THESIS

REAL-TIME, REMOTELY CONTROLLED, UNMANNED, SURFACE COMBATANT (RT-RCUSC) USING THE INTERNET

by

Floyd Bailey
&
Carl Robbins

September, 1997

Thesis Advisor: Luqi

Approved for public release; distribution is unlimited.
### REPORT DOCUMENTATION PAGE

**1. AGENCY USE ONLY (Leave blank)**

**2. REPORT DATE**
September 1997

**3. REPORT TYPE AND DATES COVERED**
Master's Thesis

**4. TITLE AND SUBTITLE**
REAL-TIME, REMOTELY CONTROLLED, UNMANNED, SURFACE COMBATANT (RT-RCUSC) USING THE INTERNET

**5. FUNDING NUMBERS**

**6. AUTHOR(S)**
Floyd Bailey & Carl Robbins

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Naval Postgraduate School
Monterey, CA 93943-5000

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for public release; distribution is unlimited.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT (maximum 200 words)**
This thesis was developed in response to the Navy's goal to reduce staffing levels aboard surface combatants. The thesis describes the computers, peripherals, and communication networks that make a Real-Time, Remotely Controlled, Unmanned, Surface Combatant (RT-RCUSC) possible using wire and wireless Internet connections and protocols.

A Command and Control (C2) model was developed for using the rapid prototype methodology. The C2 model collected latency data which was analyzed to determine the feasibility of a RT-RCUSC.

Sixteen experiments using latency times were designed to determine the viability of communication paths that progressively increased in distance and complexity. Variables included the use of two protocols, TCP and UDP, the use of two satellite types, geosynchronous and Low Earth Orbiting (LEO), as well as employing up to two satellites per end-to-end transmission path.

The results demonstrated that real-time control of a ship's navigation system can be performed when entries are made directly on the server PC or when using a client PC that is connected to the server PC via an Ethernet LAN. When controlling RT-RCUSC directly from the server using TCP and UDP, the mean latency time was approximately 31.4 milliseconds and 32.0 milliseconds respectively with the greatest latency time equal to 60 and 75.5 milliseconds respectively. Similarly when controlling RT-RCUSC from a client, connected to the server via a LAN, using TCP and UDP, the mean latency time was approximately 32.0 and 31.1 milliseconds respectively with the greatest latency time equal to 90 and 100.5 milliseconds respectively. Security restrictions prevented Java socket formation between the client/server interface when testing wireless paths aboard USS Coronado. The restrictions prevented us from gathering latency data for our geosynchronous satellite experiments. Low Earth Orbiting (LEO) satellite communications transmission/reception equipment failed to be provided for our evaluation. Therefore we were unable to gather latency data for our LEO wireless connection experiments.

Future research needs to focus on gathering latency data from both geosynchronous and LEO satellites for the purposes of determining the viability of a RT-RCUSC in order to determine the effectiveness of wireless communications with respect to Naval C2 systems.

**14. SUBJECT TERMS**
Real-Time, Remote Control, Unmanned, Surface Combatant, Internet, Latency.

**15. NUMBER OF PAGES**
223

**16. PRICE CODE**

**17. SECURITY CLASSIFICATION OF REPORT**
Unclassified

**18. SECURITY CLASSIFICATION OF THIS PAGE**
Unclassified

**19. SECURITY CLASSIFICATION OF ABSTRACT**
Unclassified

**20. LIMITATION OF ABSTRACT**
UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102
REAL-TIME, REMOTELY CONTROLLED, UNMANNED, SURFACE COMBATANT (RT-RCUSC) USING THE INTERNET

Floyd Bailey
B.S., University of Idaho, 1983

&

Carl Robbins
B.S., San Diego State University, 1978

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL

September 1997

Authors:

Floyd Bailey

Carl Robbins

Approved by:

Luqi, Thesis Advisor

Valdis Berzins, Second Reader

Ted Lewis, Chairman
Department of Computer Science
ABSTRACT

This thesis was developed in response to the Navy's goal to reduce staffing levels aboard surface combatants. The thesis describes the computers, peripherals, and communication networks that make a Real-Time, Remotely Controlled, Unmanned, Surface Combatant, (RT-RCUSC) possible using wire and wireless Internet connections and protocols.

A Command and Control (C2) model was developed using the rapid prototype methodology. The C2 model collected latency data which was analyzed to determine the feasibility of a RT-RCUSC.

Sixteen experiments using latency times were designed to determine the viability of communication paths that progressively increased in distance and complexity. Variables included the use of two protocols, TCP and UDP, the use of two satellite types, geosynchronous and Low Earth Orbiting (LEO), as well as employing up to two satellites per end-to-end transmission path.

The results demonstrated that real-time control of a ship's navigation system can be performed when entries are made directly on the server PC or when using a client PC that is connected to the server PC via an Ethernet LAN. When controlling RT-RCUSC directly from the server using TCP and UDP the mean latency time was approximately 31.4 and 32.0 milliseconds respectively with the greatest latency time equal to 60 and 75.5 milliseconds respectively. Similarly when controlling RT-RCUSC from a client, connected to the server via a LAN, using TCP and UDP, the mean latency time was approximately 32.0 and 31.1 milliseconds respectively with the greatest latency time equal to 90 and 100.5 milliseconds respectively. Security restrictions prevented Java socket formation between the client/server interface when testing wireless paths aboard the USS Coronado. The restrictions prevented us from gathering latency data for our geosynchronous satellite experiments. Low Earth Orbiting (LEO) satellite communications transmission/reception equipment failed to be provided for our
evaluation. Therefore we were unable to gather latency data for our LEO wireless connection experiments.

Future research needs to focus on gathering latency data from both geosynchronous and LEO satellites for the purposes of determining the viability of a RT-RCUSC in order to determine the effectiveness of wireless communications with respect to Naval C2 systems.
TABLE OF CONTENTS

I. INTRODUCTION ........................................................................................................ 1
   A. PROBLEM STATEMENT .................................................................................. 1
   B. RAPID PROTOTYPING .................................................................................. 3
   C. METHODOLOGY & DELIVERABLES .............................................................. 3
   D. THESIS ORGANIZATION ............................................................................ 4

II. BACKGROUND ...................................................................................................... 5
   A. TASKING ........................................................................................................ 5
      1. Today’s Military .......................................................................................... 6
         a. UAVs ........................................................................................................ 6
         b. Naval Surface Combatants .................................................................... 10
      2. The Future Navy… now to the year 2,000 .................................................. 12
      3. The Future Navy… the year 2,000 and beyond ........................................... 14
      4. More on Stealth… ....................................................................................... 17
      5. Impact of Information Technology for the 21st Century (IT-21) on the RT-
         RCUSC Design ......................................................................................... 20
      6. Internet Connections Aboard Surface Combatants Today ......................... 21
      7. Internet Connections Aboard Surface Combatants in the Future ............... 22

   B. JUSTIFICATION ............................................................................................. 23

   C. GOALS .......................................................................................................... 24

   D. REAL-TIME SYSTEMS ............................................................................... 24
LIST OF FIGURES

Figure 1. USS Yorktown, CG-48, the Smart Ship Test Platform ...................... 2

Figure 2. Predator UAV ........................................................................... 6

Figure 3. Global Hawk UAV ................................................................... 7

Figure 4. Dark Star UAV ........................................................................ 8

Figure 5. Outrider UAV .......................................................................... 9

Figure 6. The former USS Decatur (DDG 31), now the Self Defense Test Ship (SDTS) ................................................................. 10

Figure 7. Arsenal or Maritime Fire Support Demonstrator Ship ................. 13

Figure 8. CVX ....................................................................................... 17

Figure 9. Sea Shadow ........................................................................... 18

Figure 10. Sea Wraith Stealth Corvette .................................................... 19

Figure 11. Current Shipboard Internet Architecture .................................. 22

Figure 12. Server or Direct Control .......................................................... 27

Figure 13. Shipboard Client Control .......................................................... 27

Figure 14. Ashore Command Center Control ......................................... 28

Figure 15. Local Afloat Command Center Control ................................... 29

Figure 16. Long Distance Afloat Command Center Control ..................... 31

Figure 17. Overall Diagram Detailing the Physical Components of the Model .... 34

Figure 18. RT-RCUSC Home Page ............................................................ 36

Figure 19. Thesis Summary Page .............................................................. 36

Figure 20. SM4.html ............................................................................... 37

Figure 21. Nav_Mod.html ....................................................................... 37
Figure 22. Rad_Mod.html ................................................................. 37
Figure 23. Wea_Mod.html ................................................................. 38
Figure 24. Class Diagram for the Client Portion of RT-RCUSC .............. 39
Figure 25. Class Diagram for RT-Server ............................................. 40
Figure 26. Histogram of Latency Frequencies for Experiment 1 (TCP) ....... 42
Figure 27. Histogram of Latency Frequencies for Experiment 1 (UDP) ...... 43
Figure 28. Histogram of Latency Frequencies for Experiment 2 (TCP) ....... 44
Figure 29. Histogram of Latency Frequencies for Experiment 2 (UDP) ...... 45
LIST OF TABLES

Table 1. Statistics for Experiment 1 (TCP) ......................................................... 42
Table 2. Statistics for Experiment 1 (UDP) ......................................................... 43
Table 3. Statistics for Experiment 2 (TCP) ......................................................... 44
Table 4. Statistics for Experiment 2 (UDP) ......................................................... 45
ACKNOWLEDGEMENT

To Dr. Luqi, we would like to express our most profound thanks for your many hours of patient guidance. This is especially true because of the challenges presented to all of us when employing the distance learning method. We wish you the best in your future endeavors.

To the Naval Research and Development activity (NRaD), which is the Research, Development, Test and Evaluation (RDT&E) Division of the Naval Command Control and Ocean Surveillance Center (NCCOSC) for supporting this distance learning program with the Naval Post-Graduate School and proving the concept that investment in employees yields substantial returns to the employer.

To Debbie, Carl’s wife, and to Carl’s daughters, Valerie and Elizabeth, thank you all for your support and patience over the past two years. In the absence of such support this journey would not have been possible.

To our parents whose sacrifices allowed us to pursue our dreams.
I.  INTRODUCTION

A.  PROBLEM STATEMENT

The traditional unpopularity of U.S. military service personnel casualties combined with the low numbers of casualties recorded during Operation Desert Storm through the employment of U.S. developed high technology has resulted in an even greater demand on that technology to support the military. Furthermore, Operation Desert Storm proved that when an objective is quickly obtained there are significantly lower levels of U.S. military service personnel casualties. High tech “smart” weapons can rapidly suppress hostilities resulting in decreased casualties. This form of rapid and successful warfare has had a tremendous impact on the U.S. citizens. As a result not only do U.S. citizens insist on their approval prior to a conflict they also demand quick, decisive and successful action with minimal casualties. Therefore politicians and senior military personnel must respond to those demands in order to gain support before they commit U.S. forces.

With the collapse of the Soviet Union and the apparent end of the Soviet Communist Threat, demands have been placed on the U.S. military to reduce its size. This combined with the additional pressures that are being placed on our government to reduce spending in order to address the enormous Federal budget deficit have resulted in tremendous decreases in military and civilian personnel, equipment (ships, planes, etc.) and research.

The U.S. military is faced with a formidable challenge. With less money the U.S. military must resolve hostilities quickly, with a minimum of casualties, using fewer people and less equipment. In short, the U.S. military is being asked to do more with less.

The most obvious option is to employ the technology that is presently available. Coincidentally this is the objective of the current Smart Ship Project. The Smart Ship Project is an integrated product team established in November 1995 in the Naval Surface Warfare Center in Maryland for pursuing the following charter: "Develop, evaluate, and
select solutions to demonstrate that reductions in the crew's workload for a surface combatant can be achieved. Solutions will involve the application of available technology, changes to personnel and manpower policies, and changes to any other policies or procedures which drive shipboard manpower requirements. Implement these solutions using a pilot program on an operational ship and evaluate the ship's ability to maintain readiness and accomplish its mission. Identify specific billets which can ultimately be eliminated.” [Ref. 1] [Fig. 1]

![USS Yorktown, CG-48, the Smart Ship Test Platform](image)

Figure 1. USS Yorktown, CG-48, the Smart Ship Test Platform
(Courtesy, Office of the Secretary of Defense Public Affairs)

This thesis takes the Smart Ship goal to the limit by exploring the ultimate solution in reducing the staffing requirements aboard a Navy ship. This thesis describes the computers, peripherals, and communication networks that make a Real-Time, Remotely Controlled, Unmanned, Surface Combatant, RT-RCUSC (pronounced “RT-RUCKUS”) possible using wire and wireless Internet connections and protocols. The communications network used to support this system is described.

We contend that it is not only possible but practical to develop a RT-RCUSC. By eliminating all personnel from a naval combatant the number of U.S. casualties will be
reduced even if the combatant is disabled or destroyed. This approach is the ultimate solution in safeguarding against shipboard casualties should the combatant be placed in harms way.

Therefore the purpose of this thesis is to present the concept of a Real-Time Remotely Controlled Unmanned Surface Combatant (RT-RCUSC). RT-RCUSC is our contribution to the challenges that face today's U.S. Navy by posing a possible solution to the demands listed above. Our RT-RCUSC is an alternative to the traditional approach of sending both men and ships in harms way. Of course the byproduct of substituting U.S. military personnel with technology is that we significantly reduce the opportunity for casualties within our forces.

B. RAPID PROTOTYPING

Prototyping of hardware has long been accepted in the engineering discipline yet it remains relatively unused in software development. By applying the rapid prototyping methodology to our RT-RCUSC we can develop a model that will allow us to evaluate the feasibility of the RT-RCUSC without having to specify the entire system or develop all of the code [Ref. 2-5]. Lacking the need for detailed design, the RT-RCUSC development and evaluation can be accomplished very quickly through the use of iterations of the model where each iteration can be tested, verified and analyzed.

C. METHODOLOGY & DELIVERABLES

The RT-RCUSC was developed by researching the existing technology and previous applications of unmanned, remotely controlled devices, defining an overall architecture, specifying a subsystem, rapidly prototyping the software, integrating the hardware and software elements, testing and reporting on the results.

The deliverables are an executable prototype, source code, and this thesis write-up which evaluates the effectiveness of the model. The model is available for review by any
of the DOD and C4ISR stakeholders, with the intention of performing subsequent refinements as well as the substitution of actual classified parameters.

D. THESIS ORGANIZATION

Chapter II provides sufficient background information to make this thesis a stand-alone document. Chapter III describes the design of the network architecture and the strategy behind the communications employed by the model. Chapter IV provides the details of the RT-RCUSC model including a report of the experiment. Finally, Chapter V provides the summary and conclusion of the research along with a discussion of follow on research.
II. BACKGROUND

A. TASKING

Reducing staffing levels aboard surface combatants, while at the same time maintaining a high level of effective readiness and responsiveness to hostile and/or threatening conditions, is the goal of present and future Navy projects. These projects include the Self Defense Test Ship (SDTS), Arsenal or Maritime Fire Support Demonstrator Ship, Surface Combatant-21 (SC-21) and CVX. In response to this goal we accepted the task to evaluate the effectiveness of the Internet in controlling surface combatants remotely. The use of the Internet, combined with the use of military and/or commercial communication satellites, provides a path that makes remote control possible. However the demand for real-time reaction and guaranteed delivery of operator initiated commands brings into the question the ability of the Internet, using wireless satellite communications, to control a surface combatant remotely. Real-time requirements are of paramount importance to a surface combatant in several areas such as track detection and reporting, threat evaluation, weapon’s assignment and air control.

A real-time Command and Control (C2) system is required in order to demonstrate the ability of the Internet to remotely control an unmanned surface combatant. In addition the C2 system must gather latency data to determine the feasibility of this approach. Therefore a model C2 system will be developed using the rapid prototype methodology for the purposes of this evaluation. The latency data will be analyzed in order to determine the feasibility of using the Internet for real-time, remote control of a naval surface combatant.

It is important to review the current and future military systems that make use of wireless communications in order to appreciate the level of technology that either exists or is planned. A summary of wireless military systems is provided in the subsequent sections.
1. Today's Military

Today the U.S. military utilizes wireless communications to control Unmanned Aerial Vehicles (UAVs) and a surface ship. Below is a summary of UAVs and surface ship that are controlled remotely.

a. UAVs

Presently the Air Force is employing the Pentagon's most advanced unmanned spy plane, known as the Predator, which is designed to give commanders an immediate picture of troop and weapons movements on the ground. [Ref. 6][Fig. 2] The latest versions of the aircraft have radar-imaging systems enabling it to "see" through cloud cover. It is controlled by a pilot on the ground who guides the 27-foot-long plane to a target several hundred miles away, from where it can transmit images via satellite to ground commanders. It can stay on station over a target for as long as 24 hours at an altitude of 23,000 feet.

![Figure 2. Predator UAV](Image)

(Courtesy, Office of the Secretary of Defense Public Affairs)
Additionally, the Air Force is now developing another UAV. It is the Global Hawk high flying UAV, intended to provide military commanders with high-resolution, real-time imagery of large geographic areas. [Ref. 7][Fig. 3]

![Global Hawk UAV](image)

**Figure 3. Global Hawk UAV**

(Courtesy, Office of the Secretary of Defense Public Affairs)

Global Hawk is part of the Defense Advanced Research Projects Agency (DARPA) and Defense Airborne Reconnaissance Office's (DARO) High Altitude Endurance (HAE) UAV program, which is being pursued as an advanced concept technology demonstration.

The UAV is intended to operate in low-to-moderate threat environments in which it will be able to survey 40,000 square nautical miles in a single day from altitudes upwards of 65,000 feet.

According to the Pentagon, “For a typical mission, the Global Hawk system can fly to a target area 3,000 nautical miles away at 65,000 feet, and stay airborne for 24 hours collecting data before returning”. Its long endurance capability permits the vehicle to view and track critical mobile targets for long periods. Potential alternative
payloads include signals-intelligence sensors, foliage-penetration radars and communications relay packages.

Global Hawk will be complemented by the Darkstar UAV being developed for the Air Force by Lockheed Martin and Boeing. [Ref. 7][Fig. 4] Darkstar will be optimized for high threat environments, as it contains low observable, or "stealth," characteristics.

Figure 4. Dark Star UAV
(Courtesy, Lockheed Martin Corporation)

"Outrider" is the U.S. Army's UAV specialized to support Army intelligence gathering. Outrider is an advanced concept technology demonstration, aimed at getting a tactical spy drone fielded without going through the often lengthy conventional procurement process. [Ref. 8][Fig. 5]
The implication so far is that UAV’s support only reconnaissance missions; however, UAV’s can also provide additional services.

Miniature UAV’s are being developed that are the size of an adult’s hand. [Ref. 9] These Micro Aerial Vehicles (MAV) can deliver ammunition, survey targets and inspect the inside of military buildings. The U.S. Defense Advanced Research Projects Agency (DARPA) is studying the feasibility of MAV’s which measure about 15 cm (six inches) across. According to DARPA, "The sensors for these types of vehicle, if not here today, are within reach technologically and they represent a significant driver to want to build something small". MAVs are expected to be particularly useful in urban warfare where they could be employed to carry messages and carry out surveillance.

UAV’s are being considered as a low cost cruise missile defense system. The UAV would be outfitted with sensors and a kill mechanism. [Ref. 10]
b. Naval Surface Combatants

Besides UAV's and MAV's, there is a naval surface ship that has been developed that can operate, unmanned, using remote control.

Originally decommissioned in 1983, the former USS DECATUR (DDG 31) was selected for conversion to a test ship in 1988. [Ref. 11][Fig. 6] Now known as the Self Defense Test Ship (EDDG-31), or SDTS, the ship is designed, primarily, for unmanned operation on the Pacific Missile Test Range. The SDTS can be piloted remotely, and its systems can be operated remotely, thereby eliminating the safety constraints which were required in previous testing. The ship is controlled remotely by the Weapons Division, Naval Air Warfare Center, Point Mugu, CA. The combat systems installed aboard SDTS are controlled remotely by the Port Hueneme Division, Naval Surface Warfare Center, Port Hueneme, CA.

Figure 6. The former USS Decatur (DDG 31), now the Self Defense Test Ship (SDTS) (EDDG-31)
(Courtesy, U.S. Navy, via NavPhoto Archives)
A crew can also go aboard to pilot the ship and operate the installed systems and equipment whenever necessary to meet test objectives.

During typical operations, air launched threats and/or surface launched threats will attack the SDTS. The system under test, whether a new combat system configuration or an individual element (sensors, weapon systems, etc.) will respond to these threats to defend the ship.

Multiple weapons attacking the SDTS can be detected by multiple sensors and engaged by multiple defensive systems. With SDTS the T&E community has the opportunity to perform the kind of realistic, integrated, synergistic testing which cannot be accomplished aboard a manned vessel. The SDTS can also be utilized for testing when tied to the dock, and in cooperative test efforts with systems installed at the PHD NSWC Surface Warfare Engineering Facility (SWEF).

With the ship pier-side, SDTS is a floating laboratory available for use without the in port scheduling problems associated with a manned vessel. Interoperability issues can be addressed in a shipboard environment, system operational computer programs can be verified, and hardware related problems can be resolved.

Underway the vessel presents two distinct operating modes: manned and unmanned. Under either condition the vessel is available on a dedicated basis, and testing will not be preempted for routine shipboard operations, training evolutions or higher priority missions.

Remotely monitored and/or controlled systems include: Mk 23 Target Acquisition System, Mk 57 NATO SEASPARROW (Dual Directors), Combat Systems Remote Control System, Hulk Integrated Target System (Ship Remote Control), Two Wire Automatic Remote Sensing Evaluation System, Ordinance Magazines, AN/SLQ-32 ESM, Close In Weapons System (CIWS), and RAM System.

Additional examples of unmanned, remotely controlled devices can be found in the areas of space and ocean exploration. The point is the technology is well proven and universally accepted.
2. The Future Navy... now to the year 2,000

The Arsenal or the Maritime Fire Support Demonstrator Ship concept is an outgrowth of the Navy's shift in focus from the open ocean to the littoral. [Ref. 12][Fig. 7] It is fully consistent with "Forward from the Sea," and proposes an innovative means to provide more decisive, responsive, and varied naval support to the land battle. Through concentration of massive firepower, continuous availability and application of netted targeting and weapons assignment, the Arsenal Ship concept would supplement the programmed force of carriers and Tomahawk-capable combatants and submarines. The ship would be specifically tailored to meet the heavy support challenge in the opening days of conflict, without having to surge non-deployed surface ships and submarines from the United States.

The Arsenal Ship, along with other forward deployed naval and joint forces, will most likely be the key to successful introduction as well as early employment of ground forces. Initially operating under the control and umbrella of regularly deployed Aegis combatants, the Arsenal Ship concept envisions providing the Unified Commanders-in Chief (CinC) improved capability to halt or deter an invasion, and if necessary, help enable the build-up of coalition land-based air and ground forces to achieve favorable conflict resolution. With a current vision of no more than a six-ship force, the Arsenal Ships will be stationed continuously forward, always available for rapid movement upon receipt of even the most ambiguous or limiting strategic warning. Much like our maritime pre-positioning force, the Arsenal Ship proposal calls for the ships to remain on station in support of a Unified CinC for indefinite periods without dependence on host nation support or permission.

Proponents say that the Arsenal Ship can help win big wars faster with even fewer U.S. casualties than occurred in Desert Storm. The ships would provide the Navy the ability to participate, perhaps decisively, in the first hours of a war without having to wait days or weeks for the Marines, Air Force and Army to roll in from outside the theater. It is believed that the sooner and harder the Navy can strike an enemy the faster the war will be blunted or even won.
The Arsenal ship will have as many as 750 vertical launch tubes packed with Tomahawks and other smart missiles. During war, the Arsenal Ships would serve as floating missile magazines.

Figure 7. Arsenal or Maritime Fire Support Demonstrator Ship
(Courtesy, Office of the Secretary of Defense Public Affairs)

There will be no combat information center aboard the Arsenal Ship. All targeting, mission planning, command and decision functions will be made remotely from other ships, planes or ground stations. Air Force, Army or Navy controllers miles away, not the ship's crew, will target and fire its missiles. [Ref. 13] All of the Arsenal Ship's weapons should be able to be fired from remote locations, such as Arleigh Burke or Ticonderoga-class ships, aircraft carriers, or even Air Force planes and Army or Marine ground controllers. As few as 25 male and female sailors -- but no more than 100 -- will man the highly automated ship.

As envisioned, the Arsenal Ship's mission is to:

1. Deliver hundreds of smart missiles deep into enemy territory, smashing, slowing and stopping enemy tank and armored columns days before more Navy, Marine, Air Force and Army units can arrive from outside the combat theater.
2. Fire missiles able to intercept and destroy enemy ballistic missiles like the Scud and its more modern and destructive cousins.

3. Provide back-up air defense missiles to the fleet that will be launched by other ships and circling U.S. warplanes.

The Arsenal Ship defenses will be largely passive including a stealthy, radar evading design. Its survival will largely depend on destroyers and cruisers who will act as its defensive eyes and ears. Radar absorbing material and angular features will deflect enemy radar and be the ships' key design features. The ships might also be able to ballast down almost to the water-line to avoid visual or radar detection. Furthermore, if targeting can be done by other units using well-established downlink procedures, the ship need not even have much in the way of sensors, so stealth can be incorporated almost to the point of semi-submergence.[Ref. 14]

The emphasis on the Arsenal Ship is to act as a 'bridge' to the SC 21 next generation 'family of ships'.

3. The Future Navy... the year 2,000 and beyond

Future Navy warships, along with their combat, hull, mechanical and electrical systems, will be designed as a single integrated weapons system. [Ref. 15]

The "Surface Combatant 21" or SC21 approach attempts to accelerate the progress made in previous systems integration efforts (i.e. AEGIS) through what is called horizontal or total ship engineering (TSE). This revolutionary process regards the ship as a "system of systems" and provides built-in flexibility needed for insertion of future technologies.

SC21 is intended to be a family of ships which when combined will provide for the mission capabilities addressed in the Mission Need Statement (MNS)."

The SC21 MNS, approved by the Joint Requirements Oversight Council (JROC) in 1994, serves as the foundation for guiding 21st-century surface combatant design, research, development and acquisition program decisions, service and joint doctrine and cooperative efforts with U.S. allies.
The specific mission of SC-21, addressed in the MNS, is to carry the war to the enemy through offensive operations by:

1. Being able to launch and support precision strike weapons and to provide firepower support for amphibious and other ground forces.

2. Protecting friendly forces from enemy attack through the establishment and maintenance of battlespace dominance against theater missile, air, surface and subsurface threats.


In accordance with the MNS, SC21 must employ a TSE architectural approach that optimizes life cycle costs and performance; minimizes operating conflicts; permits rapid equipment upgrades; allows computational and communication resources to keep pace with commercial technology; and provides the capability to fight even if damaged.

The TSE approach is expected to promote commonality among ship classes and make maximum use of open systems and modular design in the ship's infrastructure, while accounting for emerging technologies during the developmental phase. Significant reductions in personnel requirements also are expected to be realized through automation.

Finally, the MNS requires SC-21 meet specific operational constraints. Some of these include:

1. Being fully functional in all environments, such as heavy weather, in the presence of electromagnetic, nuclear, biological and chemical contamination, and/or shock effects from nuclear and conventional weapon attack.

2. Providing helicopter and unmanned aerial vehicle (UAV) landing and hangaring facilities and ammunition storage for operational support of multi-mission armed helicopters.

3. Integrating with other U.S. Navy, Marine Corps, joint and allied forces in combined, coordinated operations. Joint goals for standardization and interoperability with the widest number of weapon and sensor systems will be achieved to the maximum feasible extent.
4. Embarking Special Operations Forces when required for selected missions.
5. Transiting through the Panama Canal.

Capitalizing on recent advances in technology is of critical importance to SC-21 design efforts. Some specific areas/technologies SC21 expects to incorporate include: missile defense and cooperative engagement capabilities (TBMD/CEC); power projection capabilities; passive defensive (survivability) capabilities, such as stealth design or low radar cross section reduction, signal intercept exploitation and acoustic signature reduction; and command, control, communications, computers and intelligence (C4I) capabilities.

There will be an integrated satellite network, a data link network and a fire control network that will enable SC-21 combatants to communicate directly to the shooter ashore.

Although not specifically designed to be unmanned certain requirements will be imposed to reduce crew size, thereby increasing the ship’s dependence on technology. Although unlikely to be specified as a remotely controlled unmanned surface combatant it may be possible to specify this capability as a causality backup.

The CVX project also intends to bring present and future technology together in an effort to produce an Aircraft Carrier designed to meet the challenges of the future by relying heavily on emerging technology. [Ref. 16][Fig. 8]
Figure 8. CVX
(Courtesy, Office of the Secretary of Defense Public Affairs)

4. More on Stealth...

Naval vessels always have had the distinct disadvantage of being quite visible and therefore vulnerable to attack and making all ships more stealthy, even aircraft carriers, is not new. [Ref. 17] Therefore making RT-RCUSC stealthy would become essential.

The United States has led the way since 1984 with its Lockheed-built Sea Shadow, a prototype patrol craft with its sides slanted upward at various angles to deflect radar. [Fig. 9] As was mentioned earlier the U.S. Arsenal, a concept ship that may one day serve as an unmanned, remote-controlled missile platform in the Persian Gulf that may also include a semi-surface profile.
France built stealth features into its new Lafayette class of frigates, whose hulls have diamond-like facets to thwart enemy radar and are coated with special radar-absorbent paint. Sweden has a similarly angular experimental patrol craft built of fiber-reinforced plastic.

Britain’s Vosper Thornycroft’s design of a 377-foot Sea Wraith stealth corvette, an anti-submarine patroller, whose multifaceted hull bears a striking resemblance to the stealth bomber, creates radar disturbances to throw oncoming missiles off-course, and its hull is designed to confuse enemy forces trying to track the vessel by radar and figure out what it is. [Fig. 10] On enemy screens, it's designed to have the "radar signature" of a small fishing boat.
Figure 10. Sea Wraith Stealth Corvette

(Courtesy, Vosper Thornycroft)

Its most intriguing feature is actually rather low-tech: nozzles that can generate a mist from the sea, hiding the ship from infrared search beams and masking hot spots that incoming heat-seeking missiles would try to sniff out.

Project Cougar, a model of a similar stealth ship design, was unveiled by BAeSEMA, a subsidiary of British Aerospace. The 311-foot steel vessel would target customers in the Pacific Rim region, where Singapore, Indonesia, Taiwan and others are building their fleets, the company said.

Designed for high-speed engagement in rough seas, that ship is propelled by jets of water to keep it quiet. Among its high-tech equipment: decoy launchers designed to seduce incoming enemy missiles away from the ship.
5. Impact of Information Technology for the 21st Century (IT-21) on the RT-RCUSC Design

Recently CINCPACFLT and CINCLANTFLT released a joint message concerning the development and implementation of IT-21. [Appendix A] To date they have provided IT-21 hardware/software implementation standards for programs that will be installing information systems on Fleet Units/Bases.

IT-21 is intended to support information superiority which has been determined to be the foundation of Joint vision 2010 battlefield dominance, as well as the war fighting vision for each service. IT-21 is a Fleet driven re-prioritization of C4I programs of record to accelerate the transition to a PC based tactical support war fighting network. The inactivation of the current DOD messaging system (AUTODIN) by Dec 99, with no planned navy infrastructure replacement, mandate the rapid implementation of this war fighting network.

The DOD Joint Technical Architecture (JTA) and Defense Information Infrastructure Common Operating Environment (DII COE) provide DOD with the Automated Information Systems (AISs) guidance required to take the navy into the 21st century. This convergence of solutions, problems and guidance provides the impetus to establish minimum Navy AIS standards at this time.

The IT-21 naval message defines all of the PC hardware and software that has been deemed acceptable. The list includes Windows NT, MS Exchange, MS Office 97, 32 bit operating systems, high resolution displays, mass storage, ATM backbone LANs with at least 100 mb/s etc. The IT-21 standards represent front end market technology, are dynamic in nature, and will continue to be closely linked to Commercial tends. The standards are intended to be minimum standards and will be updated periodically.

Therefore RT-RCUSC must assure compliance with the IT-21 message in order to gain Fleet acceptance.
6. Internet Connections Aboard Surface Combatants Today

Nearly every U.S. Navy ship today has connectivity to the Internet which provides E-Mail services and Internet access to shipboard personnel as well as offering ship unique Home Pages to land based “surfers” who may query a particular ship.

A shipboard user can access the Internet through a PC which is connected, via a router to a Timeplex Multiplexer (MUX) for message decomposition and scheduling which then passes the signal to the ship’s satellite ultra-high frequency (UHF) transmitter/receiver equipment. The signal is then received by a satellite which forwards the signal to a land based UHF transmitter/receiver. The UHF transmitter/receiver is located at one of many ground stations. In this case the signal is received at a Naval Communications and Telecommunications Area Management Station (NCTAMS) and is recomposed through the NCTAMS’s Timeplex MUX. From this point the signal is passed to one or more routers and is then allowed to pass onto the Internet. [Fig. 11]
NCTAMS = Naval Communications and Telecommunications Area Management Station

Figure 11. Current Shipboard Internet Architecture

7. Internet Connections Aboard Surface Combatants in the Future

A project by Bill Gates, chairman of Microsoft Corp., and Craig McCaw, a pioneer in cellular telephones, was awarded to Boeing Co. to coordinate the building of the Teledesic project. [Ref. 18] Teledesic would be used to build an "Internet in the sky" that would use hundreds of low-orbiting satellites to transmit data and conversations all over the world.

The plans call for Teledesic to begin high-speed two-way service—in which video and graphics would appear instantly on the computer screen—in the year 2002.
It is intended to bring closer an era of "personal" satellite communications, in which people can travel anywhere—even into the most remote regions of the world—and use a satellite telephone or link a computer to the Internet with no wires attached.

Such a system is intended to support the data rates and data volumes (bandwidth) necessary to allow for real-time, remote control, of an unmanned surface combatant.

ORBCOMM has recently deployed two Low Earth Orbiting satellites, the first in their constellation, which will provide capabilities similar to the proposed system underdevelopment by the Teledesic project. ORBCOMM has provided us two sets of equipment that will allow us to evaluate latency of LEOs.

At a recent symposium held at Stanford University entitled, “The Acceleration of World Wide Wireless Communications” companies such as Globalstar, ICO Global Communications, Bellcore, Telesis Technologies Laboratory, Netro, Hybrid, Hughes and NEC represent large companies that are all investing in the growth and improved services that can be obtained from wireless networks that support high speed integrated services in local, metropolitan and wide area environments. [Ref. 19]

B. JUSTIFICATION

Remote control of a surface combatant has been demonstrated using the SDTS, however its control is limited to the area contained within the Pacific Missile Range, specifically within the area covered by the line-of-sight RF transmitter towers located at San Nicholas Island, Point Hueneme and from Point Conception.

Internet connectivity offers over-the-horizon communications to naval surface combatants through satellite ultra-high frequency (UHF) links.

Therefore our stated hypothesis is that, “Real time, remote control, of an unmanned surface combatant is possible using the Internet when latency times are small enough to support real time control”. In order to test our hypothesis our research will employ the use of a model that we will build, which will collect latency times from both geosynchronous and Low Earth Orbiting (LEO) satellites. The data will be gathered, analyzed, and used for comparing the performance of the two types of satellites. Finally
we will interpret the results and determine if real-time, remote control, of an unmanned surface combatant is feasible using the Internet. Incidentally in addition to the requirement of small latency times in order to support the real time remote control of an unmanned surface combatant we also realize the necessity for the requirement of sufficient bandwidth. However our efforts are limited to the latency aspect.

Referring to the Acceleration of World Wide Wireless Communications symposium it is obvious that improvements in world wide communications networks are of critical importance to a large vendor contingent. Therefore it is very likely that we do not understand the full impact of our research as the investment in world wide wireless communications is so substantial, and the amount of scientific data so scarce, that companies making these investments are likely to view this data closely.

Given these sets of circumstances we will attempt to produce a model that tests our hypothesis.

C. GOALS

The goal of the research is to perform an analysis of the real-time remote control of a surface combatant using the Internet and provide the following specifics:

1. A simplified model of the system.
2. A description of the system goals hierarchy and the functions it must perform.
3. Performance constraints on the system
4. Implementation constraints on the system.
5. Resource constraints for the development project.
6. The specification of the external interfaces of the major components.

D. REAL-TIME SYSTEMS

Real time combat system processes are those that require an event driven, deterministic response in action and reaction time, regardless of the system’s state.
(loading, process state, process time, and number of processes). [Ref. 20] Defined priority level, interrupt driven computer program architectures are usually required to provide event driven, deterministic responses vice multi-process computer program environments, which rely on cyclic status checking to determine if an event has occurred that requires a reaction. These architectures rely on machine speed to keep the cycle within reaction time requirements. In general, they are dependent on loading, process state, process time, and number of processes.

RT-RCUSC will require event driven, deterministic response in action and reaction time regardless of system state (loading, process state, process time, and number of processes).

Reasons for the real time requirement are guaranteed reaction time and known response. Related factors are state of the system at event time (loading, processes, etc.), data/action sequencing and need for graceful degradation.

Examples of typical events are radar contact detection, ID threat determination and engagement decision.

Timeliness issues are critical in real time systems. In particular two specific time intervals are of interest, service time and latency. [Ref. 21]

Service time is the net time taken to compute a response to a given event and is primarily a function of the algorithm used in the computation and is often deterministic and predictable.

Latency is the interval between the time of occurrence of an input and the time at which it starts being serviced. It takes into account a combination of different delays and is generally unpredictable.

Our model implements a small navigation algorithm with the specific goal of minimizing service time thereby allowing us to test 5 different configurations that RT-RCUSC may operate under so that latency data can more easily be obtained. Once the latency data has been obtained, and service time has been subtracted, we will perform an analysis on the latency data in order to determine the feasibility of RT-RCUSC.

In the construction of a deployable system both the service time and latency times for a given input are combined which yield the overall reaction time for that input. In
order to build effective real-time systems, especially those systems that have hard real-
time requirements such as weapons system engagements and air control, it is important
that the interval times always meets their prescribed deadline times. A hard real-time
system requires that all deadlines must be met, otherwise the system is considered
unacceptable.

E. SPECIFIC METHODOLOGY

Five specific experiments are to be conducted which will measure latency time
between command initiation (client request) and command acknowledgment (server
receipt). In accordance with the requirements set forth by the IT-21 Standards, PCs, that
use Pentium processors, will be used. Additional equipment installed in the PCs will also
meet IT-21 standards. The PCs will run under Windows NT, one a server, the other a
client. The RT-RCUSC’s home page, written in HTML and Java, as well as the
interfacing pages will be installed on the server PC and will be accessed by the client PC.
The server PC, whether simulated or actual, is intended to be installed aboard the RT-
RCUSC. Communications between the client and server will make use of existing
TCP/IP or UDP protocols and are accessed by the Internet Explorer web browser
software package. Software will be included which will capture the latency times. The 5
experimental conditions are:

1. Simulated navigation control where command initiation is entered directly at
RT-RCUSC’s server. This experiment is intended to simulate control of the RT-RCUSC
as if were being performed aboard the RT-RCUSC, directly at the RT-RCUSC’s server
PC. This is called the "server or direct control" experiment. [Fig. 12]
2. Simulated navigation control where command initiation is entered into RT-RCUSC's client PC which, through communications via the shipboard LAN, accesses the RT-RCUSC server PC. This experiment is intended to simulate control of the RT-RCUSC as if were being performed aboard the RT-RCUSC from a client PC that sends commands to the server PC. This is called the "Shipboard Client Control" experiment. [Fig. 13]
3. Simulated navigation control where command initiation is entered into a land-based command center’s client PC which passes the commands through the following path: the client’s router, the Internet, the NCTAMS’s router, the NCTAMS’s Timeplex MUX, the NCTAMS’s satellite UHF equipment, the satellite, the RT-RCUSC’s satellite UHF equipment, the RT-RCUSC’s Timeplex MUX, and ultimately to the RT-RCUSC’s server which controls the peripheral equipment. This is called the "Ashore Command Center Control" experiment. It is intended to test the use of a single satellite where one (two way) wireless connection is employed. [Fig. 14]

![Diagram of the communication system](image)

**NCTAMS = Naval Communications and Telecommunications Area Management Station**

**Figure 14. Ashore Command Center Control**

4. Simulated navigation control where command initiation is entered into a command ship’s client PC which passes the commands through the following path: the
command ship's router, the command ship's Timeplex MUX, the command ship's satellite UHF equipment, the satellite, the NCTAMS's satellite UHF equipment, the NCTAMS's Timeplex MUX, the NCTAMS's router, the NCTAMS's Timeplex MUX, the NCTAMS's satellite UHF equipment, the satellite, the RT-RCUSC's satellite SHF equipment, the RT-RCUSC's Timeplex MUX, and ultimately to the RT-RCUSC's server which controls the peripheral equipment. This is called the "Local Afloat Command Center Control" experiment. "Local" refers to the condition where the same satellite and same NCTAMS are used by both the Local Afloat Command Center and by the RT-RCUSC when they are in communication with each other. It is intended to test the use of a single satellite where two (two way) wireless connections are employed. It would demonstrate the communications path that would be used within a battle group. [Fig. 15]

NCTAMS = Naval Communications and Telecommunications Area Management Station

Figure 15. Local Afloat Command Center Control
5. Simulated navigation control where command initiation is entered into a command ship’s client PC which passes the commands through the following path: the command ship’s router, the command ship’s Timeplex MUX, the command ship’s satellite UHF equipment, the satellite, a NCTAMS #1’s satellite UHF equipment, NCTAMS #1’s Timeplex MUX, NCTAMS #1’s router, the Defense Communications and Telecommunications Network (DCTN), NCTAMS #2’s router, NCTAMS #2’s Timeplex MUX, NCTAMS #2’s satellite UHF equipment, the satellite, the RT-RCUSC’s satellite SHF equipment, the RT-RCUSC’s Timeplex MUX, and ultimately to the RT-RCUSC’s server which controls the peripheral equipment. The Defense Communications and Telecommunications Network (DCTN) provides a large and rapid data transfer capability through the use of dedicated commercial leased connections. This is called the "Remote Afloat Command Center Control” experiment. “Remote” refers to the condition where different satellites and different NCTAMS are used by the Remote Afloat Command Center and by the RT-RCUSC when they are in communication with each other. It is intended to test the use of a two satellites where two (two way) wireless connections are employed. It would demonstrate the communications path used for world wide control. [Fig. 16]
NCTAMS = Naval Communications and Telecommunications Area Management Station
DCTN = Defense Communications and Telecommunications Network

Figure 16. Long Distance Afloat Command Center Control

Timing data will be collected for each of the experimental conditions and the results will be analyzed in an effort to test the hypothesis.
III. NETWORK TOPOLOGY

A. NETWORK ARCHITECTURE FOR THE MODEL

Installation of a PC which performs as a server running under Windows NT, a ship unique home page can be established which provides access, through a menu of hypertext links to other web pages that would offer a user the ability to control and monitor the various systems aboard the ship. In our model we will display the RT-RCUSC home page and offer only the Navigation Control hypertext link as a demonstrable entity. Other example systems will simply respond by saying, “This Site Under Construction”.

An actual fielded version of a RT-RCUSC would require network security. Presently the use of SIPERNET by the Navy provides an adequate level of protection through the use of KG-194 encryption equipment. Although not available to us for our evaluation, future RT-RCUSC designs should include KG-194s in the construction of the network and timing delays included in the latency figures.

B. INTERNET COMMUNICATIONS STRATEGY

Figure 17 provides the overall diagram that details the specific requirements for the components of the model. The model support communications to RT-RCUSC either directly on the RT-RCUSC server, from a client that is aboard RT-RCUSC connected to the server via a LAN (not shown), from a Land Based Command Center, from a Local Afloat Command Center or from a Long Distance Afloat Command Center. Zero to two satellites are required in order to send commands to RT-RCUSC. The Land Based Command Center, the Local Afloat Command Center, the Long Distance Command Centers all require the use of the Naval Communications and Telecommunications Area Management Stations (NCTAMS) for access to the military communications satellites. In addition the Long Distance Afloat Command Center requires the use of the Defense
Communications and Telecommunications Network (DCTN) for transmission of information between NCTAMSSs.

NCTAMS = Naval Communications and Telecommunications Area Management Station
DCTN = Defense Communications and Telecommunications Network

Figure 17. Overall Diagram Detailing the Physical Components of the Model
IV. RT-RCUSC MODEL

A. THE MODEL FOR ALL DEVICES AND COMMUNICATIONS LINKS

The RT-RCUSC model is a C2 application program which makes use of the Internet for command message routing. The Java programming language was selected as the language for implementing the RT-RCUSC model as it supports the concepts as outlined by the IT-21 standards and because it supports the development of Internet software. TCP and UDP were selected as the protocols for evaluation as they are the defacto standards for two way communications between Internet processes.

1. General Description

RT-RCUSC is made up of three major components: the supporting HTML web pages, sever and client software. The HTML web pages are created using HTML files. The server and client software are contained in Java files.

2. HTML Files

There are six HTML files for the TCP and UDP versions of RT-RCUSC. Main.html file is the Home Page for RT-RCUSC which provides the user the option of viewing a summary of the thesis or executing the RT-RCUSC model. [Appendix B, Fig. 18] Ths_Sum.html file is the web page displaying the thesis summary. [Appendix C, Fig. 19] The Sm4.html web page offers the user the option of selecting one of three C2 pages: Navigation, Weapons, or RADAR. [Appendix D, Fig. 20] The Nav_Mod.html web page downloads the applet that allows the user to interface with the RT-RCUSC model. [Appendix E, Fig. 21] Web pages Rad_Mod.html and Wea_Mod.html are place holder sites to be used in the future for the implementation of the radar and weapons control pages and are not currently implemented in our model. [Appendices F and G, Figs. 22 and 23]
RT-RCUSC (UDP)

The Real-Time, Remotely Controlled, Unmanned, Surface Combatant (RT-RCUSC) is the exploration of using Internet connections and protocols for control and communication between an unmanned ship and various controlling sites. See thesis description for more information. Thesis demonstration brings up a WEB site of options that are prototypes developed in this thesis used to collect timing information.

Figure 18. RT-RCUSC Home Page

RT-RCUSC Thesis Summary

This thesis was developed in response to the Navy's goal to reduce staffing levels aboard surface combatants. The thesis describes the computers, peripherals, and communication networks that make a Real-Time, Remotely Controlled, Unmanned, Surface Combatant, (RT-RCUSC) possible using wire and wireless Internet connections and protocols.

A Command and Control (C2) model was developed using the rapid prototype methodology. The C2 model collected latency data which was analyzed to determine the feasibility of a RT-RCUSC.

Sixteen experiments using latency times were designed to determine the viability of communication paths that progressively increased in distance and complexity. Variables included the use of two protocols, TCP and UDP, the use of two satellite types, geosynchronous and Low Earth Orbiting (LEO), as well as employing up to two satellites per end-to-end transmission path.

The results demonstrated that real-time control of a ship's navigation system can be performed when entries are made directly on the server PC or when using a client PC that is connected to the server PC via an Ethernet LAN. When controlling RT-RCUSC directly from the server using TCP and UDP the mean latency time was approximately 31.4 and 32.0 milliseconds respectively with the greatest latency time equal to 60 and 75.5 milliseconds respectively. Similarly when controlling RT-RCUSC from a client, connected to the server via a LAN, using TCP and UDP, the mean latency time was approximately 32.0 and 31.1 milliseconds respectively with the greatest latency time equal to 90 and 100.5 milliseconds respectively. Security restrictions prevented Java socket formation between the client/server interface when testing wireless paths aboard the USS Coronado. The restrictions prevented us from gathering latency data for our geosynchronous satellite experiments. Low Earth Orbiting (LEO) satellite communications transmission/reception equipment failed to be provided for our evaluation. Therefore we were unable to gather latency data for our LEO wireless connection experiments.

Future research needs to focus on gathering latency data from both geosynchronous and LEO satellites for the purposes of determining the viability of a RT-RCUSC in order to determine the effectiveness of wireless communications with respect to Naval C2 systems.

Figure 19. Thesis Summary Page
RT-RCUSC Model Options

- Navigation Model
- Radar and IFF Model
- Weapons Model

Figure 20. SM4.html

RT_RCUSC SCRIPTED NAVIGATION CONTROL MODEL (UDP)

Figure 21. Nav_Mod.html

RT-RCUSC RADAR and IFF MODEL

UNDER CONSTRUCTION

Figure 22. Rad_Mod.html
3. Java Files

In addition to the HTML files, RT-RCUSC requires three Java files: RT_RCUSCHpage.java, RT_Server.java and Smart_Constants.java.

a. Java Classes for RT_RCUSCHpage.java

RT_RCUSCHpage.java is the client portion of the RT-RCUSC model and is made up seven Java classes. [Appendix I for TCP, Appendix L for UDP] The class RT_RCUSCHpage is the driver for the client. This class is responsible for instantiation of the NavControl, Net_Connection, Nav_Message, NavUpdate, Compass and Plot classes. [Fig. 24] RT_RCUSCHpage provides periodic and non periodic commands, from the client to the server, such as “START”, “STOP”, course, speed and positional changes.

The NavControl and NavUpdate classes accept and validate user entered data. Nav Control initiates and terminates the simulation program contain within RT_Server. NavUpdate processes user entered changes once the system has begun execution.

The Compass and Plot classes display the heading and position of the RT-RCUSC model. The Compass class consists of a compass rose and a needle that indicates the current heading of RT-RCUSC. The Plot class provides a historical display of RT-RCUSC’s last fifteen positions. The last position as reported by the server is at the center of the grid with numerical x and y coordinates displayed on the axis.
The **Nav_Message** class is responsible for building and parsing navigation messages. Data designated for transmission between the client and server are converted from integers into bytes for transmission through the Internet. Data received from the Internet are converted from bytes back into integers. The transmitted and converted data are used to update the Compass and Plot display panels.

![Class Diagram](image)

**Figure 24. Class Diagram for the Client Portion of RT-RCUSC**

The **Net_Connection** class is responsible for establishing a client socket and for communicating with the server program. The **Net_Connection** establishes a client socket and blocks until a socket is established. Once a socket has been established, navigation update messages are sent and received from the server. When these messages are received, the **Net_Connection** class sets a flag that is periodically checked by the **RT_RCUSC_Hpage** class indicating that another message is awaiting processing. Socket error conditions are captured by the **Net_Connection** class.

**b. Java Classes for RT_Server.java**

The **RT_Server.java** file is made up four Java classes. [Appendix J for TCP, Appendix M for UDP] The class **RT_Server** is the driver for RT-RCUSC’s server.
This class is responsible for instantiation of the Model, Net_S_Connection and Nav_Message classes. [Fig. 25] RT_Server periodically calls the Model class requesting updates of RT-RCUSC's speed, heading and position attributes. Once RT_Server receives the updates from the Model class it then sends the data to the Nav_Message class for integer to byte conversion. Following the conversion, the data are returned to RT_Server where it controls the flow of the data to Net_S_Connection. Net_S_Connection class sends the update to the client site. RT_Server also parses messages received from the client and determines whether to update the command attributes or to stop the simulation.

![Class Diagram for RT-Server](image)

**Figure 25. Class Diagram for RT-Server**

The Net_S_Connection class is responsible for establishing a server socket and for communicating with the client program. The Net_S_Connection establishes a server socket and blocks until the client establishes a connection. Once a socket has been established, navigation update messages are sent and received from the client. When these messages are received, this Net_S_Connection class sets a flag that is periodically checked by the RT_Server class indicating that another message is awaiting processing. Socket error conditions are captured by the Net_S_Connection Class.
The Model class is responsible for modeling the simulated ship's attributes such as course, speed and position. This class is called periodically from RT_Server. The position and heading attributes are used to build navigation messages.

The Nav_Message class is responsible for building and parsing navigation messages. Data designated for transmission between the client and server are converted from integers into bytes for transmission through the Internet. Data received from the Internet are converted from bytes back into integers. The transmitted and converted data are used to update the Model's class command attributes.

c. Java Constants for Smart_Constants.java

The constants used by the RT_RCUSC model are contained in file Smart_Constants.java. [Appendix K for TCP, Appendix N for UDP]

B. REPORT ON THE EXPERIMENT

Initial data collection began shortly after integration testing was complete, the start date was July 6, 1997. At that time a PC which had Microsoft NT Server installed on it was unavailable to us so we executed experiments 1 and 2 from a SUN machine (white.nosc.mil) which enables us to further debug the software. This data was not included in the thesis as it was collected on equipment that did not meet the IT-21 standard. By July 12, 1997 we had secured a PC with NT Server installed in it which allowed us to execute our first two experiments.

Appendix O contains the data for experimental condition one (Server Control using TCP). Experimental condition one for TCP yielded latency mean times of 31.4 milliseconds and the greatest latency time was 60 milliseconds. [Table 1] The histogram for this data reveals a distribution composed of two modes. [Fig. 26]

41
Table 1. Statistics for Experiment 1 (TCP)

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>31.4104</td>
</tr>
<tr>
<td>Standard Error</td>
<td>0.5580441406</td>
</tr>
<tr>
<td>Median</td>
<td>35</td>
</tr>
<tr>
<td>Mode</td>
<td>45</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>13.95103516</td>
</tr>
<tr>
<td>Sample Variance</td>
<td>194.6313821</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>-1.207254334</td>
</tr>
<tr>
<td>Skewness</td>
<td>-0.21443877</td>
</tr>
<tr>
<td>Range</td>
<td>55</td>
</tr>
<tr>
<td>Minimum</td>
<td>5</td>
</tr>
<tr>
<td>Maximum</td>
<td>60</td>
</tr>
<tr>
<td>Sum</td>
<td>19631.5</td>
</tr>
<tr>
<td>Count</td>
<td>625</td>
</tr>
<tr>
<td>Largest(1)</td>
<td>60</td>
</tr>
<tr>
<td>Smallest(1)</td>
<td>5</td>
</tr>
<tr>
<td>Confidence Level(95.0%)</td>
<td>1.095866014</td>
</tr>
</tbody>
</table>

Figure 26. Histogram of Latency Frequencies for Experiment 1 (TCP)

Appendix P contains the data for experimental condition one (Server Control using UDP). Experimental condition one for UDP yielded latency mean times of 32.0 milliseconds and the greatest latency time was also 75.5 milliseconds. [Table 2] The histogram for this data reveals a distribution composed of two modes and the possibility of the formation of a third mode. [Fig. 27]
<table>
<thead>
<tr>
<th>Statistics for Experiment 1 (UDP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean 32.01355932</td>
</tr>
<tr>
<td>Standard Error 0.80659356</td>
</tr>
<tr>
<td>Median 35</td>
</tr>
<tr>
<td>Mode 45</td>
</tr>
<tr>
<td>Standard Deviation 13.85369935</td>
</tr>
<tr>
<td>Sample Variance 191.9249856</td>
</tr>
<tr>
<td>Kurtosis -0.870757961</td>
</tr>
<tr>
<td>Skewness -0.278277581</td>
</tr>
<tr>
<td>Range 70.5</td>
</tr>
<tr>
<td>Minimum 5</td>
</tr>
<tr>
<td>Maximum 75.5</td>
</tr>
<tr>
<td>Sum 9444</td>
</tr>
<tr>
<td>Count 295</td>
</tr>
<tr>
<td>Largest(1) 75.5</td>
</tr>
<tr>
<td>Smallest(1) 5</td>
</tr>
<tr>
<td>Confidence Level(95.0%) 1.587428297</td>
</tr>
</tbody>
</table>

Table 2. Statistics for Experiment 1 (UDP)

![Histogram of Latency Frequencies for Experiment 1 (UDP)](image)

Figure 27. Histogram of Latency Frequencies for Experiment 1 (UDP)

Appendix Q contains the data for experimental condition two (Shipboard Client Control using TCP). Experimental condition two for TCP yielded latency mean times of 32.0 milliseconds and the greatest latency time was 90 milliseconds. [Table 3] The histogram for this data reveals a distribution composed of two modes and the possibility of three modes. [Fig. 28]
Table 3. Statistics for Experiment 2 (TCP)

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>32.0057377</td>
</tr>
<tr>
<td>Standard Error</td>
<td>0.659294408</td>
</tr>
<tr>
<td>Median</td>
<td>35</td>
</tr>
<tr>
<td>Mode</td>
<td>10</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>16.28337068</td>
</tr>
<tr>
<td>Sample Variance</td>
<td>265.1481608</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>-0.814801658</td>
</tr>
<tr>
<td>Skewness</td>
<td>0.172826799</td>
</tr>
<tr>
<td>Range</td>
<td>85</td>
</tr>
<tr>
<td>Minimum</td>
<td>5</td>
</tr>
<tr>
<td>Maximum</td>
<td>90</td>
</tr>
<tr>
<td>Sum</td>
<td>19523.5</td>
</tr>
<tr>
<td>Count</td>
<td>610</td>
</tr>
<tr>
<td>Largest(1)</td>
<td>90</td>
</tr>
<tr>
<td>Smallest(1)</td>
<td>5</td>
</tr>
<tr>
<td>Confidence Level(95.0%)</td>
<td>1.294766769</td>
</tr>
</tbody>
</table>

Figure 28. Histogram of Latency Frequencies for Experiment 2 (TCP)

Appendix R contains the data for experimental condition two (Server control using UDP). Experimental condition two for UDP yielded latency mean times of 31.1 milliseconds and the greatest latency time was 100.5 milliseconds. [Table 4] The histogram for this data reveals a distribution composed of two modes and the possibility of four modes. [Fig. 29]
Statistics for Experiment 2 (UDP)

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>31.1271028</td>
</tr>
<tr>
<td>Standard Error</td>
<td>0.742804415</td>
</tr>
<tr>
<td>Median</td>
<td>30</td>
</tr>
<tr>
<td>Mode</td>
<td>10</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>17.1811159</td>
</tr>
<tr>
<td>Sample Variance</td>
<td>295.1907435</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>-0.323314938</td>
</tr>
<tr>
<td>Skewness</td>
<td>0.526468684</td>
</tr>
<tr>
<td>Range</td>
<td>95.5</td>
</tr>
<tr>
<td>Minimum</td>
<td>5</td>
</tr>
<tr>
<td>Maximum</td>
<td>100.5</td>
</tr>
<tr>
<td>Sum</td>
<td>16653</td>
</tr>
<tr>
<td>Count</td>
<td>535</td>
</tr>
<tr>
<td>Largest(1)</td>
<td>100.5</td>
</tr>
<tr>
<td>Smallest(1)</td>
<td>5</td>
</tr>
<tr>
<td>Confidence Level(95.0%)</td>
<td>1.459178071</td>
</tr>
</tbody>
</table>

Table 4. Statistics for Experiment 2 (UDP)

Figure 29. Histogram of Latency Frequencies for Experiment 2 (UDP)

Coincidentally the results obtained for experiments 1 and 2 using the SUN workstation as a server were nearly identical to the results obtained when using the IT-21 compliant equipment.

In all cases the histograms presented a distribution that suggested more than one mode. This indicates that collisions had occurred and retransmission was necessary.
The operational condition of the equipment and software was confirmed using a series of informal tests on July 13, 1997 in preparation for the execution of experimental conditions 3, 4 and 5 aboard the USS Coronado using a geosynchronous satellite.

On July 14 a series of tests were executed in an effort to exercise experimental condition 3. Because USS Coronado was unable to provide us access to their SIPERNET server due to security constraints and because the ship does not have a NIPERNET server we elected to run the experiment in a configuration that placed the server on land and the controlling client aboard the ship. This configuration was opposite in physical characteristics to our defined environment. Experimental condition three was intended to support the use of an Ashore Command Center that would control RT-RCUSC by attaching a client PC from the Ashore Command Center, via the Internet, to the server PC located aboard RT-RCUSC (in this case USS Coronado). Therefore without access to a shipboard server we decided to use a shipboard client and contact our land based server. It was felt that this configuration would provide identical data sets had we been able to configure the system as originally defined.

The shipboard client immediately located our server web site (http://sanchita.nosc.mil/java/tcp/nav_mod.html) and the applets were passed and began executing successfully. However after the navigation data was entered and the “START” button pressed the client was unsuccessful in forming a socket connection back to the server. The result was that no latency data could be obtained. This was consistent for both the TCP and UDP protocols.

An investigation into the hardware connections aboard the ship proved inconclusive as the ship’s Information Services personnel felt that the problem involved either timing constraints imposed by the protocols when the protocols are used to communicate via a satellite or of because of a security check or firewall that may have been established between the client and server PCs. The IS personnel aboard Coronado indicated that if there was a firewall established aboard the ship the result would have been that we would not have reached our web site as the firewall is IP specific. Therefore the IS personnel were satisfied that no firewall was present that would have prevented us from conducting the experiment.
During the week of 14 July access to our web site was made via several other communication methods such as modems and remote Internet access through the World Wide Web and in all cases it was possible to manipulate the navigation data and collect latency data. In addition we developed another software program, “RT-Client” which was intended to execute from the client and manipulate the data on the server without the need for a web browser or HTML interface. In addition RT-Client contained logic that captured the error conditions which would allow us to determine where the fault occurred.

Upon arrival from USS Coronado data analysis was conducted on 19-20 July for the week of 14 July. In addition RT-Client was made ready for its installation aboard USS Coronado for the week of 21 July. Also on July 17 we solicited the assistance of the satellite communications group at the Naval Research and Development (NRaD) Center, San Diego, requesting access to the Navy communication satellites via an Ethernet connection. Although the Center was quite generous with their equipment, personnel and assistance bandwidth restrictions were in place by the NCTAMS eastpac because of Fleet demands which therefore eliminated this option.

On July 21 RT-Client was installed and executed aboard USS Coronado. The resultant error condition stated, “java.net.NoRouteToHostException: Host unreachable”. When referring to the documentation for Java errors this condition indicated that either a firewall was present or that a router was not working. Since access to the web site was obtained both prior to and following the RT-Client run it was assumed that the router(s) were in place and functioning, therefore ruling out the possibility of a router problem. With access to the web site uninterrupted by firewall protection it was considered questionable if security constraints played a role in denying access to the server data.

Subsequent discussions with Dr. Luqi on July 26 resulted in our generation of an E-Mail to Professor Volpano requesting assistance. Indications are that the programs have been developed properly and an explanation for the problem remains unknown.

In an attempt to perform the experiments contacts were made to colleagues in Australia, Guam and Puerto Rico. Latency times obtained from the remote execution of the program from Australia indicated a mean time of 192 milliseconds. This value is
significantly lower than that expected if communications were conducted via satellite as the propagation delay for end-to-end transit time for a geosynchronous satellite transmission is 270 milliseconds. [Ref. 22] As a result of this test we have learned that distance alone is not the proper criteria in assuring that geosynchronous satellite communications are invoked when using the Internet.

In addition to Australia, Guam and Puerto Rico we attempted to contact the USS Constellation (CV-64) through their Public Affairs Office in an effort to request their participation in our experiment, unfortunately we were unable to make contact with personnel able to assist.

On August 8, 1997 we conducted an experiment to reinforce our findings that the USS Coronado is not configured to accept Java socket connections. Indications were that when USS Coronado was in port that its connection to the Internet was via an Ethernet connection provided by a pier side cable connections. When we discovered that we could access the Internet from the ship but were prevented from making a socket connection we felt confident that it was a security setting disallowing a socket connection. Returning to NRaD we tested the pathway to USS Coronado by “pinging” a computer aboard the USS Coronado. We discovered that the latency time was 641 milliseconds and the trace indicated that the NCTAMS was included, possibly indicating that a satellite path was used. Certainly the latency time suggests a satellite path was used. Therefore we could not prove that the problem was caused by security restrictions aboard the ship or timing problems within the communications path.

To date we have been unsuccessful in gathering latency data for experiment 3 and in executing experimental conditions 4 and 5 using geosynchronous satellites.

The testing of experimental conditions 3-5 using LEO satellites became a possibility when on June 3, 1997 Mr. Jerry Neuner, of AIRINC, offered LEO satellite transmission/reception equipment for our thesis work. AIRINC is a license for the LEO satellite transmission/reception equipment built by ORBCOMM which is the company that owns and operates its own LEO satellite constellation. Mr. Neuner contacted ORBCOMM on my behalf to make the arrangements for the equipment delivery. Mr. Stan Young of ORBCOMM contacted me directly concerning the use and application of
the equipment and upon his approval the equipment was to be delivered. On July 10, I contacted Mr. Neuner to determine the status of the equipment. Mr. Neuner indicated that ORBCOMM had approved my request for the equipment but had not delivered it to AIRINC who would immediately forward it on to me. In addition Mr. Neuner, who had offered technical assistance in modifying our software in order for it to execute using the ORBCOMM equipment, requested a copy of the software. I immediately provided him copies of our program.

Since July 10 I have contacted Mr. Neuner twice by phone and have been told that the equipment is, "on its way".

Without the ORBCOMM equipment in our possession we are unable to perform our LEO satellite data collection and analysis.

Based on the logistical issues outlined above we feel that we have exhausted our options in obtaining the support necessary in order to complete all aspect of the experimental data collection necessary to bring this topic to closure.
V. SUMMARY AND CONCLUSION

A. EVALUATION OF THE MODEL

The model was found to be useful in the collection of latency timing data. The data was used to perform a variety of statistical tests and to develop histograms which were used to evaluate the timing constraints of a real-time command and control system that communicates from client to host using the Internet. The model's effectiveness was never fully realized because of restricted availability of both geosynchronous and LEO satellites.

B. UNRESOLVED ISSUES

Due to the restricted availability of both geosynchronous and LEO satellites it was not possible for us to complete experimental conditions 3, 4 and 5. At such time when those restrictions are lifted the experiments should be conducted and the data analyzed. The results of the analysis should be presented and used a guideline for the latency considerations when trying to built a C4ISR system that requires remote control through the Internet.

C. SUMMARY

Real-time, remote control of an unmanned surface combatant using the Internet is a capability that is within our present day technological grasp. It was our hope that our experimental conditions would have allowed for the collection of latency data that would have provided the critical timing constraints associated with the use of the Internet for such control. The timing constraints could then be incorporated into the architectural considerations for systems such as RT-RCUSC. Unfortunately our research fell short of it goal.
In reviewing the latency data that was collected for both experimental conditions one and two, using either TCP or UDP protocols, our results reinforce the premise that real-time control is not only possible but practical when controlling a system directly from the server or via a client connected through a Local Area Network (LAN). With latency values substantially below the 100 millisecond level constructing a real-time system is certainly achievable.

It was very disappointing for us when we came to the realization that we would be unable to collect the latency data for experimental conditions three, four and five.

Unidentified problem located within the Navy's geosynchronous satellite communications network as well as limited access to the Navy's satellite communications equipment restricted our ability to complete our geosynchronous data collection.

Logistical problems involving the release of commercial LEO satellite communications equipment eliminated our ability to gather our LEO satellite data.

The final value of this effort may offer a limited contribution to our colleagues in the discipline of Software Engineering. Yet if such an assignment's true value is judged by the enlightenment and knowledge gained by the students who have pursued this topic then it has achieved the loftiest of goals. The level of integration required in order to conduct these experiments has been a substantial challenge which has opened many new doorways both intellectually as well as interpersonally.

Certainly RT-RCUSC is possible. Social and political influences may restrict the use of a RT-RCUSC in the immediate future however based on the technological trend it is obvious that this technology will be refined and implemented with greater intensity in the near future.

Of constant concern is the question of security. Work in the areas of network security is addressing these concerns and therefore it seems reasonable to assume that the technology will develop rapidly.

Vulnerability of such systems to electronic jamming is certainly possible however communication schemes that provide encryption and frequency shifting seem likely to solve those issues.
LIST OF REFERENCES


APPENDIX A. IT-21 DEFINITION

This Naval message, distributed by the Commander's in Charge of the Pacific and Atlantic Fleets (CINCPACFLT, CINCLANTFLT), provides the Information Technology for the 21st (IT-21) hardware/software implementation standards for programs installing information systems on Fleet units/bases and provides the Fleet with guidance on maintaining existing information systems until the installation of IT-21 products.

RATUZYUW RUEOMCB9916 0891106-UUUU--RUWFOAA.
ZNR UUUU ZU RUPHSGG9842 0890945 ZU RUEOMCB6600 0891100
ZUI RUEOMCB6613 0891103
RHHJAA T JICPAC HONOLULU HI
RHNVDD T USS SUPPLY
RHOOMIU T MIWU ONE ONE THREE
RHRMCAA T NAVCOMTELSTA BAHRAIN
RUCTFOA T USCGC KEY LARGO
RUHBABA T CG THIRD MARDIV
RUHBANA T MTCC OKINAWA JA
RUHDYOK T NAVSECGRUACT YOKOSUKA JA
RUWNVAL T USS JOHN S MCCAIN
RUWNWQ T USS MCLLUSKY
R 300944Z MAR 97 ZYB PSN 077925Q30
FM CINCPACFLT PEARL HARBOR HI//N00://
TO ALPACFLT
ALLANTFLT
INFO RUENAAA/ASSTSECNAV RDA WASHINGTON DC//C411/
RUENAAA/CNO WASHINGTON DC//N00/N09/N095/N2/N4/N41/N43/N46/N6/N6B/
N8/N80/N39/N87/N88/
RUCBCLF/CINCPACFLT NORFOLK VA//N00/N66/
RUCBACM/CINCUSACOM NORFOLK VA//J00/J66/
RHHMUNA/USCINCPAC HONOLULU HI//J00/J66/
RHDLCNE/CINCUSNAVEUR LONDON UK//J00/N66/
RULSSEA/COMNAVSEASYSCOM WASHINGTON DC//N00/N08/PMS335/PMS3
RUENMED/BUMED WASHINGTON DC//N00/
RHRMDAB/RUCJNAV/COMUSNAVCENT//N00/N66/
RUCTPOA/CNET PENSCOLA FL//N00/
RUEACNP/BUPERS WASHINGTON DC//N00/
RUHEHMS/COMMARFORPAC//CG/G66/
RUCKMAA/COMMARFORLANT//CG/G66/
RULSSPA/COMSPAWARSYSCOM WASHINGTON DC//N00/N05/PMW171/PMW1766/
RUWFDOD/NAVSTKAIRWRCN FALLON NV//N00/
RULKSDF/COMNAVSECGRU FT GEORGE G MEADE MD//N00/
RULSAX/COMNAVSUPSYSCOM MECHANICSBURG PA//N00/
RUWFAF/COMNAVSECSWRCOM CORONADO CA//N00/
RULSDG/NRL WASHINGTON DC//N00/
RULSWCB/COMNAVCOMTELCOM WASHINGTON DC//N00/N36/
RUCOSOA/NAVMASSO CHESAPEAKE VA//N00/N66/
RUWFOAA/NCCOSC RDTE DIV SAN DIEGO CA/N433//
RHHMHAH/CINCPACFLT PEARL HARBOR HI/N00/
BT
UNCLAS /N05230/
ALPACFLT 008/97
MSGID/GENADMIN/CINCPACFLT/008/
SUBJ/INFORMATION TECHNOLOGY FOR THE 21ST CENTURY/
POC/D.A. STRAUB/CDR N6/CINCLANTFLT/-/TEL: 757 322-5863/
RMKS/1. THIS IS THE FIRST IN A SERIES OF JOINT CINCPACFLT AND
CINCLANTFLT MESSAGES CONCERNING THE DEVELOPMENT AND IMPLEMENTATION
OF IT-21. THIS MESSAGE PROVIDES IT-21 HARDWARE/SOFTWARE
IMPLEMENTATION STANDARDS FOR PROGRAMS INSTALLING INFORMATION
SYSTEMS ON FLEET UNITS/BASES AND PROVIDES THE FLEET WITH GUIDANCE
ON MAINTAINING EXISTING INFORMATION SYSTEMS UNTIL INSTALLATION OF
IT-21 PRODUCTS. THE IT-21 IMPLEMENTATION STANDARDS OUTLINED BELOW
ARE PROMULGATED IN ADVANCE OF DON-WIDE GUIDANCE FROM THE DON CHIEF
INFORMATION OFFICER (CIO). THE DON CIO WILL PROMULGATE DON-WIDE
STANDARDS FOLLOWING NEGOTIATION OF ENTERPRISE-WIDE NETWORK
OPERATING SYSTEMS AND APPLICATIONS.
2. BACKGROUND: INFORMATION SUPERIORITY IS THE FOUNDATION OF
JOINT VISION 2010 BATTLEFIELD DOMINANCE, AS WELL AS THE
WARFIGHTING VISION FOR EACH SERVICE. NETWORK WARFARE, ROBUST
INFRASTRUCTURE AND INFORMATION DISSEMINATION TO DISPEED FORCES
ARE KEY ELEMENTS IN ACHIEVING INFORMATION SUPERIORITY. IT-21 IS
A FLEET DRIVEN REPRIORITIZATION OF C4I PROGRAMS TO ACCELERATE THE TRANSITION TO A PC BASED TACTICAL/TACTICAL SUPPORT
WARFIGHTING NETWORK. THE INACTIVATION OF THE CURRENT DOD MESSAGING
SYSTEM (AUTODIN) BY DEC 99, WITH NO PLANNED NAVY INFRASTRUCTURE
REPLACEMENT, MANDATES THE RAPID IMPLEMENTATION OF THIS WARFIGHTING
NETWORK.
3. COMMERCIAL NETWORK OPERATING SYSTEMS (NOS) AND E-MAIL PRODUCTS
HAVE ACHIEVED FUNCTIONAL PART. THE FLEETS CANNOT CONTINUE TO
SUPPORT A MULTITUDE OF DIVERSE OPERATING SYSTEMS AND E-MAIL PRODUCTS
WITH THEIR OWN TRAINING, OPERATIONAL PROCEDURES AND TROUBLESHOOTING
REQUIREMENTS. THE DOD JOINT TECHNICAL ARCHITECTURE (JTA) AND
DEFENSE INFORMATION INFRASTRUCTURE COMMON OPERATING ENVIRONMENT
(DII COE) PROVIDE DOD WITH THE AIS SYSTEM GUIDANCE REQUIRED TO
TAKE THE NAVY INTO THE 21ST CENTURY. THIS CONVERGENCE OF SOLUTIONS,
PROBLEMS AND GUIDANCE PROVIDES THE IMPELMENTUS TO ESTABLISH MINIMUM
NAVY AIS STANDARDS AT THIS TIME. IMPLEMENTATION OF THIS POLICY
REQUIRES ALL NON-STANDARD NOS AND E-MAIL PRODUCTS BE REPLACED NLT
DEC 99.
   A. WINDOWS NT SERVER 4.0 IS THE STANDARD FLEET NOS. IT WILL
SOON BE FOLLOWED BY WINDOWS NT 5.0. WINDOWS NT SERVER 4.0 IS DII
COE COMPLIANT.
   B. MS EXCHANGE IS DESIGNATED AS THE STANDARD E-MAIL SOLUTION
FOR BOTH FLEETS TO ENSURE AN INTEROPERABLE SECURE MESSAGING SYSTEM
IS OPERATIONAL PRIOR TO AUTODIN INACTIVATION NLT DEC 99.
   C. MS OFFICE 97 IS DESIGNATED AS THE STANDARD FLEET OFFICE
SOFTWARE.
   D. EXPENDITURE OF OPERATING FUNDS TO MAINTAIN EXISTING IT-21
NONCOMPLIANT NOS AND APPLICATIONS SHALL BE THE ABSOLUTE MINIMUM
NECESSARY TO MEET OPERATING REQUIREMENTS UNTIL IT-21 NOS/SOFTWARE
IS INSTALLED EVEN IF TEMPORARY LAN DEGRADATION OCCURS. SOFTWARE
REQUIREMENTS DRIVE HARDWARE STANDARDS. HARDWARE AND SOFTWARE PURCHASED TODAY MUST BE CAPABLE OF MEETING MISSION REQUIREMENTS THROUGH THE YEAR 2000.

4. CINCPACFLT AND CINCLANTFLT ARE ACTIVELY WORKING WITH OPAV ON IT-21 FUNDING AND IMPLEMENTATION PLANS. IN GENERAL, AFLOAT IT-21 IMPLEMENTATION WILL BE LINKED TO DEPLOYING BATTLEGROUPS AND ASHORE IT-21 WILL BE IMPLEMENTED IN A PHASED APPROACH. SPECIFIC IMPLEMENTATION SCHEDULES WILL BE PROMULGATED AT A LATER DATE. CINCPACFLT AND CINCLANTFLT ARE TRANITIONING TO WINDOWS NT 4.0, MS EXCHANGE AND MICROSOFT OFFICE 97. THIS ENVIRONMENT CANNOT BE OPTIMIZED WITHOUT 32 BIT OPERATING SYSTEMS, HIGH RESOLUTION Displays AND MASS STORAGE. ATM BACKBONE LANS WITH AT LEAST 100 MBS (TCP/IP) TO THE DESKTOP PC WILL BE INSTALLED ON ALL SHIPBOARD LANS, FLEET HEADQUARTERS (CPF, CLF, TYCOMS, GROUPS AND SQUADRONS) AND SHOULD BE INSTALLED IN THOSE SHORE ACTIVITIES THAT SUPPORT TACTICAL OPERATIONS. THIS WILL THEN ALLOW TRANSITION TO ATM TO THE-DESKTOP PC WHEN THE ATM TECHNOLOGY MATURES.

5. SYSTEM COMMANDS AND PROGRAM MANAGERS:
   A. NTCSS WILL BECOME THE IT-21 PROGRAM O RECORD FOR INSTALLATION OF BOTH SECRET AND UNCLASSIFIED LANS ONBOARD COMMISSIONED SHIPS. NTCSS (ATIS/SNAP III) LANS INSTALLED FROM THIS POINT ON WILL HAVE AN ATM BACKBONE, 100 MBS (FAST ETHERNET) TO THE DESKTOP PC AND THE HARDWARE/SOFTWARE OUTLINED AT THE END OF THIS MESSAGE. THE MIGRATION OF NTCSS LANS TO HIGHER CAPACITY LANS WILL REDUCE THE NUMBER OF PC'S DELIVERED DURING INITIAL INSTALLATION. THE TRADE-OFF OF QUANTITY FOR FRONT END PC'S IS REQUIRED TO SUPPORT JV-2010 AND AUTODIN INACTIVATION.
   B. SPAWAR IS WORKING WITH NAVSEA TO ENSURE THAT LANS INSTALLED DURING NEW CONSTRUCTION MEET THE IT-21 REQUIREMENTS.
   C. APPLICATION PROGRAM MANAGERS SUCH AS JMCIS, NSIPS, TAMS, AND GCSS SHOULD MIGRATE CURRENT APPLICATIONS TO THE DII COE WITH AN IMMEDIATE OBJECTIVE OF OBTAINING PC WORKSTATION ACCESS TO ALL APPLICATION DATA ON AN ENTERPRISE LAN.
   D. PROGRAMS INSTALLING INFORMATION SYSTEMS (NEWNET, SMARTLINK, SMARTBASE, TELEMEDICINE, ETC.) MUST INSTALL COMPONENTS IN FLEET ACTIVITIES THAT MEET IT-21 STANDARDS AND PROVIDE INTEROPERABILITY THROUGHOUT THE WARFIGHTING NETWORK.

6. TYCOMS AND THIRD ECHELON COMMANDS SHALL ENSURE THAT:
   A. SHIPS AND ACTIVITIES INSTALLING NEW LANS, UNDERGOING SIGNIFICANT LAN UPGRADES OR THOSE ACTIVITIES WITH STAND ALONE PC'S SHALL INSTALL IT-21 HARDWARE AND SOFTWARE. NEW OR REPLACEMENT SHIPBOARD AND SHORE BASED TACTICAL LANS SHOULD HAVE AN ATM BACKBONE WITH AT LEAST 100 MBS (FAST ETHERNET) TO THE PC.
   B. SHIPS AND ACTIVITIES WITH EXISTING LANS, WHICH REQUIRE REPLACEMENT OF UNSERVICEABLE HARDWARE, SORT OF A FULL NETWORK UPGRADE, SHALL INSTALL HARDWARE WHICH MEETS IT-21 STANDARDS. THE NEW EQUIPMENT MAY NOT BE COMPATIBLE WITH THE EXISTING LAN HARDWARE. CINCPACFLT AND CINCLANTFLT BELIEVE THAT ALL AUTOMATED INFORMATION SYSTEMS (AIS) PROCURED MUST BE COMPATIBLE WITH THE IT-21 LAN STANDARDS EVEN IF TEMPORARY LAN DEGRADATION OCCURS. THERE IS ONLY SUFFICIENT FUNDING TO DO IT RIGHT THE FIRST TIME.

7. THE IT-21 STANDARDS BELOW REPRESENT FRONT END MARKET TECHNOLOGY, ARE DYNAMIC IN NATURE, AND WILL CONTINUE TO BE CLOSELY LINKED TO COMMERCIAL TENDS. THE STANDARDS LISTED BELOW ARE INTENDED TO BE
MINIMUM STANDARDS AND WILL BE UPDATED PERIODICALLY.
A. IT-21 LAN:
   (1) AFLOAT LAN STANDARDS - ATM FIBER BACKBONE, 100 MBPS
   (2) ASHORE TACTICAL AND HEADQUARTERS COMMAND CENTER STANDARD
       (CPF, CLF, TYCOMS, GROUP AND SQUADRON COMMANDS) - ATM BACKBONE, 100
       MBPS (FAST ETHERNET) TO THE PC.
   (3) ASHORE TACTICAL SUPPORT COMMAND STANDARDS (BASES) - ATM
       =20
       =20
       =20
PAGE 03 RUHPSGG9916 UNCLAS
BACKBONE, 100 MBPS (FAST ETHERNET) TO THE PC.
   (4) METROPOLITAN AREA NETWORKS (MAN) SHOULD BE CAPABLE OF
       SUPPORTING AT LEAST OC-3 (155MBPS).
B. IT-21 SOFTWARE:
   - WINDOWS NT 4.0/5.0 WORKSTATION
   - MS OFFICE 97 PROFESSIONAL (WORD 97, POWERPOINT 97, EXCEL 97, S
     ACCESS 97)
   - IBM ANTI VIRUS (NAVY LICENSE, AVAIL FROM NAVCIRT)
   - MS BACK OFFICE CLIENT
   - MS OUTLOOK 97
   - MS EXCHANGE 5.0
   - MS IMAGE COMPOSER
C. IT-21 DATABASES. RELATIONAL DATABASES THAT CAN SUPPORT WEB
   TECHNOLOGY IAW THE COE (ORACLE, SYBASE, SQL SERVER, ACCESS, ETC.)
   WILL BE USED TO SUPPORT DATA REQUIREMENTS AND APPLICATION
   DEVELOPMENT. ALL PROCESS ENGINEERING INITIATIVES THAT RESULT IN
   DESIGN/REDESIGN OF A DATA COLLECTION/CAPTURE SYSTEM MUST USE COE
   COMPLIANT RELATIONAL DATABASE MANAGEMENT SYSTEMS (RDBMS) SOFTWARE.
   THIS REQUIREMENT IS PROVIDED TO ENSURE RDBMS INITIATIVES USE COTS
   APPLICATION SOFTWARE. FOR ADDITIONAL INFORMATION ON RELATIONAL
   =20
   =20
   =20
PAGE 04 RUHPSGG9916 UNCLAS
DATABASES CONTACT CDR SANDY BUCKLES, CPF N67, COMM/DSN (808)
474-6384, NIPRNET U67@CPF-EMH.CPF.NAVY.MIL.
D. MINIMUM IT-21 PC CAPABILITIES: CPF CAN CURRENTLY PURCHASE
   THE IT-21 STANDARD PC WITH SOFTWARE FOR $3250.00 - $3579.00 -
   SEE PARA 7(H) AND 7(I).
   - 200 MHZ PENTIUM PRO CPU
   - 64 MB EDO RAM
   - 3.0 GB HARD DRIVE
   - 3.5 INCH FLOPPY DISK DRIVE
   - 8X IDE CD-ROM
- DUAL PCMCIA/PC CARD READER
- PCI VIDEO W/2MB RAM
- 17 INCH MONITOR (1280 X 1024)
- POINTING DEVICE (TRACKBALL OR MOUSE)
- SOUNDBLASTER (COMPATIBLE) AUDIO CARD WITH SPEAKERS KEYBOARD
- CPU COMPATIBLE 100 MBPS FAST ETHERNET NIC

E. STANDARD IT-21 LAPTOP WORKSTATION: APPROXIMATELY $5300 -
SEE PARA 7(H).
- 150 MHZ PENTIUM
- 32 MB EDO RAM
  =20
  =20
  =20

PAGE 05 RUHPSSG9916 UNCLAS
- 12.1 IN SVGA ACTIVE MATRIX COLOR DISPLAY
- 2.1 GB IDE HDD
- 6X INTERNAL CD-ROM
- MODEM, PCMCIA SLOTS, NIC CARD
- SMART LITHIUM BATTERY

F. IT-21 NT FILE SERVER FOR DIRECTORY NETWORK SERVICE:
APPROXIMATELY $26K - SEE PARA 7(H). THESE ARE MINIMUM
SPECIFICATIONS. NEEDS OF THE SPECIFIC NETWORK WILL DictATE
REQUIREMENTS.
- DUAL 166 MHZ PENTIUM CPU
- 512K SECONDARY CACHE MEMORY- 256 MB RAM
- TWO 4 GB SCSI HDD
- ONE 6 GB DAT DRIVE
- ONE 3.5 INCH FLOPPY DISK DRIVE
- 6X SCSI CD-ROM
- DUAL PCMCIA/PC CARD READER
- 2 DPT SCSI III CACHING CONTROLLERS (SMARTCACHE 4)
- PCI VIDEO W/2MB RAM
- 17 INCH MONITOR (1280 X 1024)
- POINTING DEVICE (TRACKBALL OR MOUSE)
  =20
  =20
  =20

PAGE 06 RUHPSSG9916 UNCLAS
- KEYBOARD
- TWO CABLETRON CPU COMPATIBLE ATM NIC CARDS
- ANTEC DUAL POWER SUPPLY CASE (HOT SWAPPABLE)

G. IT-21 FILE SERVER/APPLICATION SERVER: APPROXIMATELY $26K -
SEE PARA 7(H). SAME AS IT-21 NT FILE SERVER FOR DIRECTORY NETWORK
SERVICE WITH THE FOLLOWING CHANGES:
- CHANGE HDD RQRTM TO FIVE 4 GB DRIVES
- CHANGE DAT TO 18 GB.

H. PRICES FOR PC TECHNOLOGY ARE CONSTANTLY CHANGING AND CAN
VARY GREATLY DEPENDING ON METHOD OF PROCUREMENT. FOR EXAMPLE, ON
28 MAR 97 AN IT-21 PC PURCHASED DIRECTLY FROM A VENDOR COSTS $3643.
GOVERNMENT RATE FOR SMALL PURCHASES (LESS THAN TEN) IS $3579.
A BULK PROCUREMENT (MORE THAN SEVENTY-FIVE) COSTS $3250. THE ABOVE
PRICES INCLUDE SHIPPING. BULK PROCUREMENTS SHOULD BE MADE THROUGH
THE TYPE COMMANDERS WHEN APPROPRIATE. MR. RICK KOOKER, CPF N65,
COMM/DSN:(808) 474-5882, NIPRNET: U65@CPF-EMH.CPF.NAVY.MIL.
AVAILABLE TO ASSIST TYCOMS WITH AIS PROCUREMENT ISSUES.

1. AS NETWORK COMPUTER TECHNOLOGY EVOLVES SOME COMMANDS MAY BE
ABLE TO TRANSITION TO NETWORK COMPUTERS. WHEN CONSIDERING
INSTALLATION OF NETWORK COMPUTERS, TOTAL NETWORK COST MUST BE

PAGE 07 RUHPSSG9916 UNCLAS
EVALUATED. NETWORK COMPUTERS HAVE NOT MATURER SUFFICIENTLY TO
IMPLEMENT THEM IN FLEET PLATFORMS AT THIS TIME.

8. WAIVER REQUESTS FROM THE ABOVE STANDARDS SHOULD BE SUBMITTED
DIRECTLY TO THE RESPECTIVE CPF/CLF N6. POINTS OF CONTACT ARE AS
FOLLOWS:

A. CINCLANTFLT: CDR DEBRA STRAUB AT COMM (757) 322-5863,
NIPRNET: U6@CLF.NAVY.MIL

B. CINCPACFLT: CDR MIKE SCOTT AT COMM (808) 474-7860,
NIPRNET:U6@CPF-EMH.CPF.NAVY.MIL//

BT#9916
APPENDIX B.  MAIN.HTML

<HTML>
<HEAD>
<TITLE>RT-RCUSC Home Page (TCP) </TITLE>
</HEAD><BODY>
<P align=center><b>
<font size=+3>
RT-RCUSC (TCP)
</font></b></P>

    The Real-Time, Remotely Controlled, Unmanned, Surface Combatant (RT-RCUSC)
    is the exploration of using Internet connections and protocols for control
    and communication between an unmanned ship and various controlling sites. See
    <a href="THS_SUM.HTML">thesis description</a> for more information.
    <a href="sm4.html">Thesis demonstration</a> brings up a WEB site of options
    that are prototypes developed in this thesis used to collect timing information.
</BODY>
</HTML>
APPENDIX C. THS_SUM.HTML

<HTML>
<HEAD>
<TITLE>Thesis Abstract Page (TCP)</TITLE>
</HEAD><BODY>
<P align=center><b>
<font size=+2>
RT-RCUSC Thesis Abstract (TCP)
</font></b></p>
<APPLET CODE="Thesis_Sum.class" WIDTH=980 HEIGHT=1000></APPLET>
</BODY>
</HTML>
APPENDIX D. SM4.HTML

<HTML>
<HEAD>
<TITLE> Option Selection Page (TCP) </TITLE>
</HEAD>
<BODY>
<P align=center><h>
<font size=+2>
RT-RCUSC Model Options (TCP)
</font></h></p>
<ul>
<li><a href="nav_mod.html">Navigation Model</a></li>
<li><a href="rad_mod.html">Radar and IFF Model</a></li>
<li><a href="wea_mod.html">Weapons Model</a></li>
</ul>
</BODY>
</HTML>
APPENDIX E. NAV_MOD.HTML

<HTML>
<HEAD>
<TITLE>Navigation Module Page (TCP) </TITLE>
</HEAD><BODY>
<P align=center><b>
<font size=+2>
RT-RCUSC SCRIPTED NAVIGATION CONTROL MODEL (TCP)
</font></b></p>
<APPLET CODE="RT_RCUSCHpage.class" WIDTH=980 HEIGHT=500><APPLET>
</BODY>
</HTML>
APPENDIX F.  RAD_MOD.HTML

<html>
<head>
<title>RADAR & IFF Module Page (TCP)</title>
</head>
<body>
<p align=center><b>
<font size=+2>
RT-RCUSC RADAR & IFF CONTROL MODEL (TCP)
</font></b></p>

UNDER CONSTRUCTION
</body>
</html>
APPENDIX G.  WEA_MOD.HTML

<HTML>
<HEAD>
<TITLE>Weapons Module Page (TCP) </TITLE>
</HEAD><BODY>
<P align=center><b>
<font size=+2>
RT-RCUSC WEAPONS CONTROL MODEL (TCP)
</font></b></p>
    UNDER CONSTRUCTION
</BODY>
</HTML>
import java.awt.*;

// Class Thesis_Sum contains a summary of the thesis.
// This class is executed when thesis description is
// selected from the root web page.
public class Thesis_Sum extends java.applet.Applet
{
    public void init()
    {
        setLayout(new GridLayout(2,1,10,10));
        Panel panel1 = new Panel();
        add(panel1);
        final String summary =
        "This thesis was developed in response to the Navy’s goal to reduce staffing levels \n" +
        "aboard surface combatants. The thesis describes the computers, peripherals, and \n" +
        "communication networks that make a Real-Time, Remotely Controlled, \n" +
        "Unmanned, Surface Combatant, (RT-RCUSC) possible using wire and wireless \n" +
        "Internet connections and protocols. \n" +
        "A Command and Control (C2) model was developed using the rapid prototype \n" +
        "methodology. The C2 model collected latency data which was analyzed to \n" +
        "determine the feasibility of a RT-RCUSC. \n" +
        "Sixteen experiments using latency times were designed to determine the \n" +
        "viability of communication paths that progressively increased in distance and \n" +
        "complexity. Variables included the use of two protocols, TCP and UDP, the use \n" +
        "of two satellite types, geosynchronous and Low Earth Orbiting (LEO), as well as \n" +
        "employing up to two satellites per end-to-end transmission path. \n" +
        "The results demonstrated that real-time control of a ship’s navigation system \n" +
        "can be performed when entries are made directly on the server PC or when using \n" +
        "a client PC that is connected to the server PC via an Ethernet LAN. When \n" +
        "controlling RT-RCUSC directly from the server using TCP and UDP the mean \n" +
        "latency time was approximately 31.4 and 32.0 milliseconds respectively with \n" +
        "the greatest latency time equal to 60 and 75.5 milliseconds respectively. Similarly \n" +
        "when controlling RT-RCUSC from a client, connected to the server via a LAN, \n" +
        "using TCP and UDP, the mean latency time was approximately 32.0 and 31.1 \n" +
        "milliseconds respectively with the greatest latency time equal to 90 and 100.5 \n" +
        "milliseconds respectively. Security restrictions prevented Java socket formation \n" +
        "between the client/server interface when testing wireless paths aboard the USS \n" +
        "Coronado. The restrictions prevented us from gathering latency data for our \n" +
        "geosynchronous satellite experiments. Low Earth Orbiting (LEO) satellite \n" +
        "communications transmission/reception equipment failed to be provided for our \n" +
        "evaluation. Therefore we were unable to gather latency data for our LEO \n" +
        "wireless connection experiments. \n" +
        "Future research needs to focus on gathering latency data from both \n" +
    }
"geosynchronous and LEO satellites for the purposes of determining the viability \n" +
"of a RT-RCUSC in order to determine the effectiveness of wireless communications \n"with respect to Naval C2 system. \n"; panel1.add(new TextArea(summary,20,60));
} // End of init

} //End of Thesis_Sum
APPENDIX I. RT_RCUSCHPAGE.JAVA FOR TCP

// RT_RCUSCHPage.java is the client portion of the RT_RCUSC software. It is made
// up of seven classes. RT_RCUSCHPage is the driver of the client program and is
// responsible for instantiation of the NavControl, Net_Connection, Nav_message
// NavUpdate, Compass and Plot classes.

import java.awt.*;
import java.net.*;
import java.io.*;

// Class RT_RCUSCHPage - Main class of this applet.
// RT_RCUSCHPage is the driver of the client portion
// of RT_RCUSC program.
// It's responsible for instansiation of the four
// supporting objects - NavControl, NavUpdate, Compass,
// and Plot.
/** RT_RCUSCHPage.java
 * @version 1.1
 * @author Floyd Bailey
 * Implementation for RT_RCUSCHPage main control
 */

/**
 * Constructor for class RT_RCUSCHPage.
 */

public class RT_RCUSCHPage extends java.applet.Applet
implements Runnable, Smart_Constants
{
    // VARIABLES for RT_RCUSCHPage
    // init_nav - NavControl object declaration
    // update_nav - NavUpdate object declaration
    // compass_update - Compass object declaration
    // plot_update - Plot object declaration
    // ship_model - Model object declaration
    // first_time - Boolean variable indicating whether
    // - initial course and speed was entered
    // i - Integer variable used for exception
    // - handler
    // InitThread - Thread object declaration
    // MAX_ARRAY - Integer constant used to determine the
    // - size of array
    // point_history - Array of Point used to be last MAX_ARRAY
    // - x,y position of RT_RCUSC
    // temp_point - Temporary Point variable
    // array_index - Integer variable indicating the number of
    // - elements in point_history
    // ship_info - Integer array used to update RT_RCUSC
    // - position parameters
    // cmd_info - Integer array used to update RT_RCUSC

75
// command parameters
// ret_value
// of methods returning boolean type
// panel1
// Instansiation of Java Panel class

NavController init_nav;
NavUpdate update_nav;
Compass compass_update;
Plot plot_update;
Nav_Message nav_msg; // Class to handle building and parsing nav
Net_Connection net_con; // Class to handle network connection
static boolean first_time = true;
int i = 0;
Thread InitThread;
static Point[] point_history = new Point[MAX_ARRAY];
Point temp_point, temp_point2;
static int array_index = 0;
static int[] ship_info = new int[4];
static int[] cmd_info = new int[3];
static int update_trigger = 0; //Used to trigger update every 2 sec
boolean ret_value;
String host;
static byte[] msg_buffer = new byte[MAX_MESSAGE_LENGTH];
byte[] rec_buffer = new byte[MAX_MESSAGE_LENGTH];
Panel panel1 = new Panel();
byte record_type;

/** Method init() of class SmartShip - This method instansiates objects
 * of classes NavUpdate, Compass, Plot and NavUpdate.
 */
public void init()
{
    host = this.getCodeBase().getHost();
    init_nav = new NavController(this, "LAT", "LON", "X_POS", "Y_POS", "COURSE", "SPEED", 0, 0, 0, 0, 0, 0);
    update_nav = new NavUpdate(this, "COURSE", "SPEED", host, 0, 0, 0);
    compass_update = new Compass(this);
    plot_update = new Plot(this);
    setBackground(Color.white);
    setLayout(new GridLayout(2, 2, 10, 10));
    add(init_nav);
    add(update_nav);
    add(compass_update);
    add(plot_update);
    //Get address of applet's home
    //end of init
}

/** Method insets(int,int,int,int) of class SmartShip - This method
 * determines the spaces between the four panels - init_nav,
 * update_nav, compass_update and plot_update.
 */
**/  
public Insets insets()  
{  
    return new Insets(10,10,10,10);  
}  
   // end of insets

/** Method start() of class SmartShip - This method instansiates thread  
*  InitThread and starts it. This thread updates the panel and takes  
*  inputs from the user.  
*/  
public void start()  
{  
    if (InitThread == null)  
    {  
        InitThread = new Thread(this);  
        InitThread.start();  
    }  
    // End of == null  
}  
   // End of start

/** Method stop() of class SmartShip - this method stops thread  
*  InitThread and deinstansiates it.  
*/  
public void stop()  
{  
    if (InitThread != null)  
    {  
        InitThread.stop();  
        InitThread = null;  
        net_con.stop();  
        // Disconnect connection  
    }  
    // End of != null  
}  
   // End of stop

/** Method run() of class SmartShip - This method is the main loop of the  
*  class. This is where data and control is exchanged between the five  
*  supporting objects - init_nav, update_nav, compass_update, plot_update  
*  and ship_model.  
*/  
public void run()  
{  
    while (true)  
    {  
        // Loop forever  
        if (init_nav.running())  
        {  
            if (first_time)  
            // Get SmartShip initial speed and course  
            {  
                try  
                {  
                    ret_value = init_nav.init_info(ship_info);  
                    nav_msg = new Nav_Message(ship_info[0],ship_info[1],ship_info[2],ship_info[3]);  
                }  
            }  
        }
    }
nav_msg.get_nav_msg(msg_buffer); //Get msg to send
net_con = new Net_Connection(InetAddress.getByName(host));
net_con.send_msg(msg_buffer); //Send out init position to server
    first_time = false;
} catch (Exception e)
{
    System.out.println(e);
} // End of catch
} // End of first_time
else // SmartShip initial data already entered
{
    if (update_trigger >= 23)
        // Get SmartShip command speed, course and acceleration update
    {
        update_trigger = 0;
        cmd_info[0] = 25; // Speed
        cmd_info[1] = cmd_info[1] + 3; // Course
        if (cmd_info[1] > 359)
            cmd_info[1] = 0;
        nav_msg.update_msg(cmd_info); // Nav_msg class builds update msg
        nav_msg.get_nav_msg(msg_buffer); // Get msg to send
        net_con.send_msg(msg_buffer); // Send msg to server
    } // End of if update_command
    if (net_con.data_receive()) // Has server sent info
    {
        net_con.clear_data_receive();
        net_con.get_message(rec_buffer);
        record_type = rec_buffer[0]; // Parse server info
        switch (record_type)
        {
            case NAV_TYPE :
                nav_msg.get_ship_info(rec_buffer, ship_info);
                rec_buffer[REC_SUB_TYPE_POS] = NAV_STAT;
                net_con.send_msg(rec_buffer); // Send msg back to server for timing
                break;
            default:// Shouldn't be getting here
                // End of switch logic
        } // End of if data_receive
        if (array_index < MAX_ARRAY)
        {
            temp_point = new Point(ship_info[0], ship_info[1]);
            point_history[array_index] = temp_point;
            array_index++;
        } // End of if array not full
    else
        temp_point2 = point_history[MAX_ARRAY - 1];
        for (int for_index = (MAX_ARRAY - 1); for_index > 0; for_index--)
        {
            temp_point = point_history[for_index - 1];
            point_history[for_index - 1] = temp_point2;
            temp_point2 = temp_point;
        } // End of for loop

temp_point = new Point(ship_info[0], ship_info[1]);
point_history[MAX_ARRAY - 1] = temp_point;
} // End of else array not full yet
ret_value = plot_update.plot_array(point_history, array_index);
repaint();
compass_update.repaint();
plot_update.repaint();
} // End of else Smart ship data already entered
try
{
    Thread.sleep(100); // One tenth of second delay between position updates
    update_trigger++;
} // End of try
catch (InterruptedException e) {} // Sleep exception
} // End of if running
repaint();
} // End of while (true) loop
} // End of run

/** Method update(NavControl) of class RT_RCUSC */
void update(NavControl Nav_in)
{
    // Future code can go here
} // End of update NavControl

/** Method update(NavUpdate) of class RT_RCUSC - TBD */
void update(NavUpdate Nav_in)
{
    // Future code can go here
} // End of update NavUpdate

/** Method paint(Graphics) */
public void paint(Graphics g)
{
    g.setColor(Color.red);
    g.drawString("RUCUS", 0, 60);
}

/** Class NavControl - Supporting class of RT_RCUSCHpage. */
* It's responsible for initial SmartShip data *
/

class NavControl extends Panel
{
    // VARIABLES for class NavControl
    // running - Boolean variable which indicates if object is
    // running
    // lat - TextField variable to output latitude data to screen
    // lon - TextField variable to output longitude to screen
    // x_pos - TextField variable to output x position to screen
    // y_pos - TextField variable to output y position to screen
    // course - TextField variable to output course to screen

79
// speed - TextField variable to output speed to screen
// r_lat - Integer variable for current latitude of SmartShip
// r_lon - Integer variable for current longitude of SmartShip
// r_x_pos - Integer variable for current x position of SmartShip
// r_y_pos - Integer variable for current y position of SmartShip
// r__cource - Integer variable for current SmartShip heading
// r_ispeed - Integer variable for current SmartShip speed
// MAX_SPEED - Constant maximum speed of SmartShip
// MIN_SPEED - Constant minimum speed of SmartShip
// MAX_CRS - Constant maximum allowable compass input
// MIN_CRS - Constant minimum allowable compass input
// MIN_X - Constant minimum allowable x axis value
// MAX_X - Constant maximum allowable x axis value
// MIN_Y - Constant minimum allowable y axis value
// MAX_Y - Constant maximum allowable y axis value
// MAX_LAT - Constant maximum allowable latitude value
// MIN_LAT - Constant minimum allowable latitude value
// MAX_LON - Constant maximum allowable longitude value
// MIN_LON - Constant minimum allowable longitude value
// start_button - Constant String value for push button button
// stop_button - Constant String value for push button label
// outerparent - Variable used to declare RT_RCUSCHpage as parent
// to NavControl

static boolean running = false;
static TextField lat,lon,x_pos,y_pos,course,speed;
static int r_lat = 0;
static int r_lon = 0;
static int r_x_pos = 0;
static int r_y_pos = 0;
static int r__cource = 0;
static int r_ispeed = 0;
final int MAX_SPD = 45;
final int MIN_SPD = -20;
final int MAX_CRS = 360;
final int MIN_CRS = 0;
final int MIN_X = -1024;
final int MAX_X = 1024;
final int MAX_Y = 1024;
final int MIN_Y = -1024;
final int MAX_LAT = 90;
final int MIN_LAT = -90;
final int MIN_LON = -180;
final int MAX_LON = 180;
final String start_button = "START";
final String stop_button = "STOP";
RT_RCUSCHpage outerparent;

/** NavControl is the Constructor for class NavControl. It takes 12
 * parameters 6 of type String and 6 parameters of double. NavControl
 * is where initial ship's attributes are entered. The user also enters the start
 * and stop commands here.
 * @param parm1 string label for latitude of RT_RCUSC.
 * @param parm2 string label for longitude of RT_RCUSC.
 */
* @param parm3 string label for x position of RT_RCUSC.
* @param parm4 string label for y position of RT_RCUSC.
* @param parm5 string label for course of RT_RCUSC.
* @param parm6 string label for speed of RT_RCUSC.
* @param flt1 initial Latitude of RT_RCUSC.
* @param flt2 initial Longitude of RT_RCUSC.
* @param flt3 initial x position of RT_RCUSC 0.
* @param flt4 initial y position of RT_RCUSC 0.
* @param flt5 initial course of RT_RCUSC.
* @param flt6 initial speed of RT_RCUSC.
*
NavController(RT_RCUSCPage target, String parm1, String parm2, String parm3,
        String parm4, String parm5, String parm6, double flt1, double flt2,
        double flt3, double flt4, double flt5, double flt6)
{
      this.outerparent = target;
      setLayout(new GridLayout(7,2,10,10));  //Sets up a grid of 7 rows and 2 columns

      // Convert float parameters to string
      lat = new TextField(String.valueOf(flt1),6);
      lon = new TextField(String.valueOf(flt2),6);
      x_pos = new TextField(String.valueOf(flt3),6);
      y_pos = new TextField(String.valueOf(flt4),6);
      course = new TextField(String.valueOf(flt5),6);
      speed = new TextField(String.valueOf(flt6),6);

      //Set edit boxes to a white background
      lat.setBackground(Color.white);
      lon.setBackground(Color.white);
      x_pos.setBackground(Color.white);
      y_pos.setBackground(Color.white);
      course.setBackground(Color.white);
      speed.setBackground(Color.white);

      //Create label and text fields
      add(new Label(parm1, Label.RIGHT));
      add(lat);
      add(new Label(parm2, Label.RIGHT));
      add(lon);
      add(new Label(parm3, Label.RIGHT));
      add(x_pos);
      add(new Label(parm4, Label.RIGHT));
      add(y_pos);
      add(new Label(parm5, Label.RIGHT));
      add(course);
      add(new Label(parm6, Label.RIGHT));
      add(speed);

      // Create and label two buttons
      add(new Button("START");
      add(new Button("STOP");

81
setBackground(Color.green);
} //End of constructor NavController

/** Method insets sets up the spacing between items in this panel - labels, text boxes
 * and buttons.
 */
public Insets insets()
{
    return new Insets(10,10,10,150);
} //End of insets

/** Method init_info loads initial latitude, longitude, course and
 * speed into an array.
 */
public boolean init_info(int info[])
{
    info[0] = r_lat;
    info[1] = r_lon;
    info[2] = r_course;
    info[3] = r_speed;
    return true;
} //End of init_info

/** Method action(Event, Object) handles selection of buttons
 * and TextFields on this panel.
 * @param evt type of event to trigger this action
 * @param arg object type causing this action
 */
public boolean action(Event evt, Object arg)
{
    if (evt.target instanceof TextField)
    {
        outerparent.update(this);
        return true;
    } //End of if true
    else if (evt.target instanceof Button)
    {
        String label = (String)arg;
        if (label.equals(start_button))
        {
            read_values();
            return true;
        } // End of if START button was selected
    else
    {

82
running = false;
    return true;
}     // STOP button was selected
}
else
    return false;
}       //End of action

/** Method running returns a boolean type indicating what
*  state this panel is in - running or not running.
*/
public boolean running()
{
    return running;
}       //End of running

/** Method read_values reads and checks for values entered by the user.
*/
public void read_values()
{
    // VARIABLES for method read_values
    // crs_string      - String variable used to store course data
    // spd_string      - String variable used to store speed data
    // xpos_string     - String variable used to store x position data
    // ypos_string     - String variable used to store y position data
    // lat_string      - String variable used to store latitude data
    // lon_string      - String variable used to store longitude data
    // crs_good        - Boolean variable indicating proper course entry
    // spd_good        - Boolean variable indicating proper speed entry
    // xpos_good       - Boolean variable indicating proper x position entry
    // ypos_good       - Boolean variable indicating proper y position entry
    // lat_good        - Boolean variable indicating proper latitude entry
    // lon_good        - Boolean variable indicating proper longitude entry
    // tmp_spd         - Integer variable used for computations
    // tmp_crs         - Integer variable used for computations
    // tmp_xpos        - Integer variable used for computations
    // tmp_ypos        - Integer variable used for computations
    // tmp_lat         - Integer variable used for computations
    // tmp_lon         - Integer variable used for computations

    String crs_string;
    String spd_string;
    String xpos_string;
    String ypos_string;
    String lat_string;
    String lon_string;
    boolean crs_good;
    boolean spd_good;
    boolean xpos_good;
    boolean ypos_good;
    boolean lat_good;
    boolean lon_good;
    int tmp_spd = r_ispeed;
    int tmp_crs = r_icourse;
    int tmp_xpos = r_x_pos;

    // Read the values entered by the user

    // Validate the values and assign appropriate values

    // Update the variables

    // End of method
}
```java
int tmp_ypos = r_y_pos;
int tmp_lat = r_lat;
int tmp_lon = r_lon;

// Next 7 instructions assigns values enter by user into String variables
crs_string = course.getText();
spd_string = speed.getText();
xpos_string = x_pos.getText();
ypos_string = y_pos.getText();
lat_string = lat.getText();
lon_string = lon.getText();

// Initialize data as valid.
crs_good = spd_good = xpos_good = ypos_good = lat_good = lon_good = true;

// Following try block parses user data and checks for correct range and type.
try
{
tmp_crs = Integer.parseInt(crs_string);
if ((tmp_crs > MAX_CRS) || (tmp_crs < MIN_CRS))
{
    crs_good = false;
    course.setBackgroundColor(Color.red);
    course.setText(String.valueOf(r_course));
} // End of failed course entry
tmp_spd = Integer.parseInt(spd_string);
if ((tmp_spd > MAX_SPD) || (tmp_spd < MIN_SPD))
{
    spd_good = false;
    speed.setBackgroundColor(Color.red);
    speed.setText(String.valueOf(r_ispeed));
} // End of failed speed entry
tmp_xpos = Integer.parseInt(xpos_string);
if ((tmp_xpos > MAX_X) || (tmp_xpos < MIN_X))
{
    xpos_good = false;
    x_pos.setBackgroundColor(Color.red);
    x_pos.setText(String.valueOf(r_x_pos));
} // End of failed x pos entry
tmp_ypos = Integer.parseInt(ypos_string);
if ((tmp_ypos > MAX_Y) || (tmp_ypos < MIN_Y))
{
    ypos_good = false;
    y_pos.setBackgroundColor(Color.red);
    y_pos.setText(String.valueOf(r_y_pos));
} // End of failed y pos entry
tmp_lat = Integer.parseInt(lat_string);
if ((tmp_lat > MAX_LAT) || (tmp_lat < MIN_LAT))
{
    lat_good = false;
    lat.setBackgroundColor(Color.red);
    lat.setText(String.valueOf(r_lat));
} // End of failed lat entry
tmp_lon = Integer.parseInt(lon_string);
```
if ((tmp_lon > MAX_LON) || (tmp_lon < MIN_LON))
{
    lon_good = false;
    lon.setBackground(Color.red);
    lon.setText(String.valueOf(r_lon));
} // End of failed lon entry
} // End of try
catch (Exception e)
{
} // End of catch
if ((crs_good == true) && (spd_good == true) && (xpos_good == true) &&
    (ypos_good == true) && (lat_good == true) && (lon_good == true))
{
    course.setBackground(Color.white);
    speed.setBackground(Color.white);
    x_pos.setBackground(Color.white);
    y_pos.setBackground(Color.white);
    lat.setBackground(Color.white);
    lon.setBackground(Color.white);
    course.setText(String.valueOf(tmp_crs));
    r_course = tmp_crs;
    speed.setText(String.valueOf(tmp_spd));
    r_speed = tmp_spd;
    x_pos.setText(String.valueOf(tmp_xpos));
    r_x_pos = tmp_xpos;
    y_pos.setText(String.valueOf(tmp_ypos));
    r_y_pos = tmp_ypos;
    lat.setText(String.valueOf(tmp_lat));
    r_lat = tmp_lat;
    lon.setText(String.valueOf(tmp_lon));
    r_lon = tmp_lon;
    running = true;
} // End of good values submitted
} // End of read_values
} // End of class NavController

/** Class NavUpdate is where command data is enter by the user 
 * after the start of a run.
 */
class NavUpdate extends Panel
{
    // VARIABLES for class NavUpdate
    // course - String variable used to store course data
    // speed - String variable used to store speed data
    // acceleration - String variable used to store acceleration data
    // update_cmd - Boolean variable indicating if command data was
    // - updated
    // outerparent - Object variable indicating that RT_RCUSCHPage is
    // - parent to this class

    85
// r_spd - Integer variable indicating current command speed
// r_crs - Integer variable indicating current command course
// r_acl - Integer variable indicating current command acceleration
// MAX_SPD - Constant integer, maximum speed for SmartShip
// MIN_SPD - Constant integer, minimum speed for SmartShip
// MAX_CRS - Constant integer, maximum course (degrees) for course
// MIN_CRS - Constant integer, minimum course (degrees) for course
// MAX_ACL - Constant integer, maximum acceleration for SmartShip
// MIN_ACL - Constant integer, minimum acceleration for SmartShip

static TextField course, speed, acceleration;
static boolean update_cmd = false;
RT_RCUSCHpage outerparent;
static int r_spd = 0;
static int r_crs = 0;
static int r_acl = 0;
final static int MAX_SPD = 45;
final static int MIN_SPD = -20;
final static int MAX_CRS = 360;
final static int MIN_CRS = 0;
final static int MAX_ACL = 6;
final static int MIN_ACL = -6;

/** Constructor for class NavUpdate - takes 6 parameters. 3 of type String and
 * 3 of type Integer.
 *@param target
 *@param parm1 string type used as a label for Course text field.
 *@param parm2 string type used as a label for Speed text field.
 *@param parm3 string type used as a label for Acceleration text field.
 *@param flt1 integer type used for course enter by user.
 *@param flt2 integer type used for speed enter by user.
 *@param flt3 integer type used for acceleration enter by user.
 */
NavUpdate(RT_RCUSCHpage target, String parm1, String parm2, String parm3,
           int flt1, int flt2, int flt3)
{
    this.outerparent = target;
    setBackground(Color.blue);
    // Next instruction sets up a grid for this panel - 4 rows by 2 columns.
    setLayout(new GridLayout(4, 2, 10, 40));
    String blank = " "; // Used to align submit button

    // Convert float parameters to string
course = new TextField(String.valueOf(flt1), 5);
speed = new TextField(String.valueOf(flt2), 5);
acceleration = new TextField(String.valueOf(flt3), 5);

    // Set TextField background to white.
course.setBackground(Color.white);
speed.setBackground(Color.white);
acceleration.setBackground(Color.white);

    // Create label and text fields
    add(new Label(parm1, Label.RIGHT));
    add(course);

    add(new Label(parm2));
    add(speed);
    add(new Label(parm3));
    add(acceleration);
add(new Label(parm2, Label.RIGHT));
add(speed);
add(new Label(parm3, Label.RIGHT));
add(acceleration);
add(new Label(blank, Label.RIGHT));
add(new Button("SUBMIT"));
}  //End of constructor NavUpdate

/** Method insets determines spaces between item in this panel - TextFields, labels
 * and button.
 */
public Insets insets()
{
    return new Insets(10,10,10,70);
}  //End of insets

/** This method get_cmd_data(int cmd_info[]) the current command data.
 * @param cmd_info[] is an integer array containing command data
 * The values are read into an array which the
 * caller can use to update SmartShip position.
 */
public boolean get_cmd_data(int cmd_info[])
{
    cmd_info[0] = r_spd;
    cmd_info[1] = r_crs;
    cmd_info[2] = r_scl;
    return true;
}  // end of get_cmd_data

/** Method action handles TextField and button action of this panel.
 * @param evt is of type Event, it's the event of this action.
 * @param arg is of type Object, it's the object of this action.
 */
public boolean action(Event evt, Object arg)
{
    if (evt.target instanceof TextField)
    {
        outerparent.update(this);
        return true;
    }  // End of if true
    else if (evt.target instanceof Button)
    {
        read_values();
        update_cmd = true;
        return true;
    }
    else
    {
        return false;
    }  // End of action

/** Method update_command() returns a boolean indicating if command
 * data has been updated.
 */
public boolean update_command()
{ return update_cmd;  
}                      // End of update_command

/** Method clear_update_cmd() set update_cmd to false. */
public void clear_update_cmd()
{
    update_cmd = false;
}                      // End of clear_update

/** Method read_values() validates and assigns command data entered by user. */
public void read_values()
{
    // VARIABLES for method read_values()
    // crs_string - String variable used to store course data
    // spd_string - String variable used to store speed data
    // acl_string - String variable used to store acceleration data
    // crs_good - Boolean variable indicating valid course entry
    // spd_good - Boolean variable indicating valid speed entry
    // acl_good - Boolean variable indicating valid acceleration
    // tmp_spd - Integer used for computation
    // tmp_crs - Integer used for computation
    // tmp_acl - Integer used for computation

    String crs_string;
    String spd_string;
    String acl_string;
    boolean crs_good;
    boolean spd_good;
    boolean acl_good;
    int tmp_spd;
    int tmp_crs;
    int tmp_acl;

    // Next 3 instruction assigns command data to String variables.
    crs_string = course.getText();     // Get text of course text box
    spd_string = speed.getText();      // Get text of speed text box
    acl_string = acceleration.getText(); // Get text of acceleration text box

    tmp_spd = r_spd;
    tmp_crs = r_crs;
    tmp_acl = r_acl;

    // Initialize valid data to true.
    crs_good = spd_good = acl_good = true;
    try
    {
        tmp_crs = Integer.parseInt(crs_string);
        if ((tmp_crs > MAX_CRS) || (tmp_crs < MIN_CRS))
        {
            crs_good = false;
            course.setBackground(Color.red);
        }
    }
    catch(NumberFormatException e)
    {
        System.out.println(e.getMessage());
    }
course.setText(String.valueOf(r_crs));
    }
    // End of failed course entry
tmp_spd = Integer.parseInt(spd_string);
if ((tmp_spd > MAX_SPD) || (tmp_spd < MIN_SPD))
{
    spd_good = false;
speed.setBackgroundColor(Color.red);
speed.setText(String.valueOf(r_spd));
    }
    // End of failed speed entry
tmp_acl = Integer.parseInt(acl_string);
if ((tmp_acl > MAX_ACL) || (tmp_acl < MIN_ACL))
{
    acl_good = false;
    acceleration.setBackgroundColor(Color.red);
    acceleration.setText(String.valueOf(r_acl));
    }
    // End of failed acceleration entry
}
    // End of try
}
    // End of catch

if (((crs_good == true) && (spd_good == true) && (acl_good == true))
{
    course.setBackgroundColor(Color.white);
speed.setBackgroundColor(Color.white);
    acceleration.setBackgroundColor(Color.white);
course.setText(String.valueOf(tmp_crs));
r_crs = tmp_crs;
speed.setText(String.valueOf(tmp_spd));
r_spd = tmp_spd;
    acceleration.setText(String.valueOf(tmp_acl));
r_acl = tmp_acl;
    }
    // End of good values submitted
}
    // End of read_values
}
    // End of class NavUpdate

/**
 * Class compass displays the current SmartShip heading.
 */
class Compass extends Canvas
{
    // VARIABLES for class Compass
    // COMPASS_X_CEN - Constant integer compass x center
    // COMPASS_Y_CEN - Constant integer compass y center
    // MAX_Y - Constant integer used for drawing needle
    // MAX_X - Constant integer used for drawing needle
    // MAX_LIT_Y - Constant integer used for drawing needle
    // degree - Double variable, current compass value
    // degs_rads - Double variable, radian value of the heading
    // x_values - Integer array used to draw needle
    // y_values - Integer array used to draw needle
    // pts - Integer variable, number of points to draw needle
    // rad_to_deg - Double variable, conversion factor of degrees to rads
    // outerparent - Variable indicating RT_RCUSCHpage as parent

89
// cos_ret - Double variable, cosine of heading
// sin_ret - Double variable, sine of heading

final int COMPASS_X_CEN = 130;
final int COMPASS_Y_CEN = 110;
final int MAX_Y = 50;
final int MAX_LIT_Y = 5;
final int MAX_X = 10;
static double degree = 30.0;
double degs_rads;
int x_values[] = {140,130,120,130,140};
int y_values[] = {110,60,110,113,110};
int pts = x_values.length;
double rad_to_deg = Math.PI / 180.0;
RT_RCUSCHpage outparent;
double cos_ret,sin_ret;

/** Constructor Compass(SmartShipHpage) creates an object of class Compass
 * @param target is of type RT_RCUSCHpage
 */
Compass(RT_RCUSCHpage target)
{
    this.outerparent = target;
    setBackground(Color.blue);
} // End of Compass

/** Method set_compass(int) sets new heading for compass.
 * @param compass_value is of type int, sets compass to new value.
 */
public boolean set_compass(int compass_value)
{
    degree = (double) compass_value;
    return true;
} // Set_compass

/** Method paint(Graphics) displays current heading.
 * @param g is of type Graphics, use to paint to screen.
 */
public void paint(Graphics g)
{
    degs_rads = degree * rad_to_deg;
    cos_ret = Math.cos(degs_rads);
    sin_ret = Math.sin(degs_rads);
    x_values[0] = (int) (Math.round(cos_ret * MAX_X) + COMPASS_X_CEN);
    y_values[0] = (int) (Math.round(sin_ret * MAX_X) + COMPASS_Y_CEN);
    x_values[1] = (int) (Math.round(sin_ret * MAX_Y) + COMPASS_X_CEN);
    y_values[1] = (int) (COMPASS_Y_CEN - Math.round(cos_ret * MAX_Y));
    x_values[2] = (int) (COMPASS_X_CEN - Math.round(cos_ret * MAX_X));
    y_values[2] = (int) (COMPASS_Y_CEN - Math.round(sin_ret * MAX_X));
    x_values[3] = (int) (COMPASS_X_CEN - Math.round(sin_ret * MAX_LIT_Y));
    y_values[3] = (int) (COMPASS_Y_CEN + Math.round(cos_ret * MAX_LIT_Y));
    x_values[4] = x_values[0];
    y_values[4] = y_values[0];
    // Next instruction instatiates a new polygon object - the needle
Polygon comp_needle = new Polygon(x_values, y_values, pts);
g.setColor(Color.white);
g.fillOval(50, 30, 160, 160);
g.drawString("360", 120, 25);
g.drawString("180", 120, 205);
g.drawString("270", 31, 116);
g.drawString("90", 212, 116);
g.setColor(Color.black);
g.drawLine(130, 30, 130, 190);
g.drawLine(50, 110, 210, 110);
g.fillPolygon(comp_needle);
} // End of paint
} // End of class Compass

/** Class Plot plots current (at grid center) and up to the last 15
 * SmartShip positions.
 */
class Plot extends Canvas
{
    // VARIABLES for class Plot
    // outerparent - Variable that indicates that RT_RCUSCHpage is parent
    // MAX_ARRAY - Constant int - maximum number of points that can be plotted
    // ship_history - Array of type points - positions of SmartShip
    // last_point - Number of positions in ship_history
    // x_center - String variable used to display x value of grid center
    // y_center - String variable used to display y value of grid center

    RT_RCUSCHpage outerparent;
    static final int MAX_ARRAY = 25;
    static Point[] ship_history = new Point[MAX_ARRAY];
    static int last_point = 0;
    String x_center, y_center;

    /** Constructor Plot(RT_RCUSCHpage) used to create an instance of class Plot.
     * @param target is of type RT_RCUSCHpage
     */
    Plot(RT_RCUSCHpage target)
    {
        this.outerparent = target;
        setBackground(Color.green);
        // End of Compass
    }

    /** Method plot_array(Point[], int) updates SmartShip positions array.
     * @param ship_hist[] of type Point array - last positions of RT_RCUSC.
     * @param last_entry of type int - is the number of positions in array.
     */
    public boolean plot_array(Point ship_hist[], int last_entry)
    {
        // VARIABLES for method plot_array
        // index - Integer variable used as array index
        // index2 - Integer variable used as array index
        // keep_looking - Boolean variable used for loop control

        int index;
int index2;
boolean keep_looping = true;
index2 = (last_entry - 1);
index = 0;
while (keep_looping)
{
    ship_history[index] = ship_hist[index2];
    if ((index2 <= 0) || (index >= (MAX_ARRAY - 1)))
        keep_looping = false;
    else
    {
        index++;
        index2--;
    }  // End of else
}  // End of while keep_looping
last_point = index;
return true;
}  // End of plot_array

/** Method paint(Graphics) plots SmartShip positions to panel. */
public void paint(Graphics g)
{
    // VARIABLES for method paint(Graphics)
    // REC_WIDTH    - Constant integer, grid width
    // REC_LENGTH   - Constant integer, grid length
    // index        - Integer variable used for array index
    // prev_x       - Integer variable used for line computations
    // prev_y       - Integer variable used for line computations
    // cur_x        - Integer variable used for line computations
    // cur_y        - integer variable used for line computations

    final int REC_WIDTH = 180;
    final int REC_LENGTH = 180;
    int index,prev_x,prev_y,cur_x,cur_y;
    prev_x = 0;
    prev_y = 0;
    g.setColor(Color.white);
    g.fillRect(50,15,REC_WIDTH,REC_LENGTH);
    g.setColor(Color.black);
    g.drawLine(140,15,140,195);
    g.drawLine(50,105,230,105);
    x_center = Integer.toString(ship_history[0].x);
    y_center = Integer.toString(ship_history[0].y);
    g.drawString(x_center,130,12);
    g.drawString(y_center,20,105);
    for (index = 0; index <= last_point; index++)
    {
        if (index > 0)
        {
            if ((Math.abs(ship_history[index].x) < REC_WIDTH) &
                (Math.abs(ship_history[index].y) < REC_LENGTH))
            {
                cur_x = ship_history[index].x - ship_history[0].x;
            }
cur_x = -1 * cur_x;
cur_y = ship_history[index].y - ship_history[0].y;
g.setColor(Color.green);
g.drawLine((140 + prev_y),(105 + prev_x),(140 + cur_y),(105 + cur_x));
g.setColor(Color.black);
prev_x = cur_x;
prev_y = cur_y;
}    // End of boundary check
}    // End of if > 0
}    // End of for loop
}    // End of paint
}    // End of class Plot

// Class Nav_Message builds and parses nav messages.
class Nav_Message implements Smart_Constants {
{
// VARIABLES for class Nav_Message
// MSG_LENGTH - Max length for nav message is 36 bytes
// NUM_OF_SHIP_PARMS - Nav message contains 4 parameters for ship
// LAT_parm_POS - Latitude parameter position in byte buffer
// LON_parm_POS - Longitude parameter position in byte buffer
// CRS_parm_POS - Course parameter position in byte buffer
// SPD_parm_POS - Speed parameter position in byte buffer
// Y_POS - Ship's y position in byte buffer
// X_POS - Ship's x position in byte buffer
// CRS_POS - Ship's course position in byte buffer
// SPD_POS - Ship's speed position in byte buffer
// C_CRS_POS - Ship's command course position in byte buffer
// C_SPD_POS - Ship's command speed position in byte buffer
// C_ACL_POS - Ship's command acceleration in byte buffer
// PARM_LENGTH - Parameter's length - 4 bytes
// ZERO_BYTE - Byte of 0's
// index2 - Used for loop variable
// ser_number - Serial number of message
// MAX_SERIAL - Largest serial number after which value is set to 0
// nav_data - Byte buffer used for storing messages
// char_digit - Character used to store a digit as a character

final static int MSG_LENGTH = 36;
final static int NUM_OF_SHIP_PARMS = 4;    // x pos, y pos, course and speed
final static int LAT_parm_POS = 4;
final static int LON_parm_POS = 8;
final static int CRS_parm_POS = 12;
final static int SPD_parm_POS = 16;
final static int X_POS = 4;
final static int Y_POS = 8;
final static int CRS_POS = 12;
final static int SPD_POS = 16;
final static int C_CRS_POS = 4;
final static int C_SPD_POS = 8;
final static int C_ACL_POS = 12;
final static int PARM_LENGTH = 4;
final static byte ZERO_BYTE = 0;
static byte[] nav_data = new byte[MSG_LENGTH];
char char_digit;

93
/** Nav_Message Constructor for class */
Nav_Message()
{
}

// Constructor with no arguments

/** Nav_Message(int,int,int,int) Constructor for class - 4 parameters */
* @param srt_lat is starting latitude of RT_RCUSC.
* @param srt_lon is starting longitude of RT_RCUSC.
* @param srt_crs is starting course of RT_RCUSC.
* @param srt_spd is starting speed of RT_RCUSC.
*/
Nav_Message(int srt_lat, int srt_lon, int srt_crs, int srt_spd)
{
    int index,index2;
    int num_length,num_128,rem_128;
    for (index = 0; index < MSG_LENGTH; index++)
    {
        nav_data[index] = ZERO_BYTE;        // Zero out message array
    }

    // Code to fill in speed field of nav message
    nav_data[SPD_PARM_POS + 3] = (byte)(0x0000007f & srt_spd);

    // Code to fill in course field of nav message
    num_128 = srt_crs / 128;
    nav_data[CRS_PARM_POS + 2] = (byte)num_128;
    rem_128 = srt_crs % 128;
    nav_data[CRS_PARM_POS + 3] = (byte)rem_128;
    nav_data[MSG_TYPE_POS] = NAV_TYPE;
    nav_data[REC_TYPE_POS] = NAV_INIT;
    nav_data[SUB_REC_POS] = ZERO_BYTE;
    nav_data[SER_RET_POS] = ZERO_BYTE;
}

}     // End of method Nav_Message

//Method update_msg updates nav parameters for class
void update_msg(int[] cmd_update)
{
    int num_of_128,remainder;
    int num_length;
    int index;
    for (index = 0; index < MSG_LENGTH; index++)
    {
        nav_data[index] = ZERO_BYTE;        // Zero out message array
    }

    nav_data[C_AQL_POS + 3] = (byte)(0x0000007f & cmd_update[2]); // Acceration
    nav_data[C_SPD_POS + 3] = (byte)(0x0000007f & cmd_update[0]); // Speed
    nav_data[C_CRS_POS + 2] = (byte)num_of_128;
    remainder = cmd_update[1] % 128;
    nav_data[C_CRS_POS + 3] = (byte)remainder;
    nav_data[MSG_TYPE_POS] = NAV_TYPE;
    nav_data[REC_TYPE_POS] = NAV_CMD;
    nav_data[SUB_REC_POS] = ZERO_BYTE;
    nav_data[SER_RET_POS] = ZERO_BYTE;

94
} // End of update_msg

// Method get_nav_msg retrieves nav message
void get_nav_msg(byte[] init_msg)
{
    int index;
    for (index = 0; index < MSG_LENGTH; index++)
        init_msg[index] = nav_data[index];
} // End of get_init_msg

// Method get_ship_info retrieves ship's position and course
void get_ship_info(byte[] rec_buf,int[] ship_info)
{
    int index,num_128,rem_128;

    // Code to fill in spd field of nav message
    ship_info[3] = 0;
    ship_info[3] = rec_buf[SPD_POS + 3];

    // Code to fill in crs field of nav message
    ship_info[2] = 0;
    num_128 = rec_buf[CRS_POS + 2];
    num_128 = num_128 * 128;
    rem_128 = rec_buf[CRS_POS + 3];

    // Code to fill in y pos field of nav message
    ship_info[1] = 0;
    num_128 = rec_buf[Y_POS + 2];
    num_128 = num_128 * 128;
    rem_128 = rec_buf[Y_POS + 3];
    ship_info[1] = num_128 + rem_128;

    // Code to fill in x pos field of nav message
    ship_info[0] = 0;
    num_128 = rec_buf[X_POS + 2];
    num_128 = num_128 * 128;
    rem_128 = rec_buf[X_POS + 3];
    ship_info[0] = num_128 + rem_128;
}

} // End of get_ship_info

// Method get_ship_init retrieves ship's initialization info
void get_ship_init(byte[] rec_buf,int[] ship_init_info)
{
    int index,num_128,rem_128;
    ship_init_info[0] = 0; // Zero out lat field
    ship_init_info[1] = 0; // Zero out lon field
    ship_init_info[2] = 0; // Zero out the course field
    ship_init_info[3] = 0; // Zero out the speed field

    // Parse out the speed field of nav message
    ship_init_info[3] = rec_buf[SPD_PARM_POS + 3];
// Parse out the course field of nav message
num_128 = rec_buf[CRS_PARM_POS + 2];
num_128 = num_128 * 128;
rem_128 = rec_buf[CRS_PARM_POS + 3];
ship_init_info[2] = num_128 + rem_128;
}

// End of get_ship_init

// Method get_ship_cmd retrieves ship's command course, speed and acceleration
void get_ship_cmd(byte[] rec_buf, int[] ship_cmd_info)
{
    int index, num_128, rem_128;
    ship_cmd_info[0] = 0;
    ship_cmd_info[1] = 0;
    ship_cmd_info[2] = 0;

    // Parse out the acceleration field of nav message
    ship_cmd_info[2] = rec_buf[C_ACL_POS + 3];

    // Parse out the speed field of nav message
    ship_cmd_info[1] = rec_buf[C_SPD_POS + 3];

    // Parse out the course field of nav message
    num_128 = rec_buf[C_CRS_POS + 2];
    num_128 = num_128 * 128;
    rem_128 = rec_buf[C_CRS_POS + 3];
    ship_cmd_info[0] = num_128 + rem_128;
}

// End of get_ship_cmd

// Method build_status_msg parses status message received from server
void build_status_msg(byte[] nav_data, int[] ship_status_info)
{
    int index, num_128, rem_128;
    int temp_value;
    / * Used to prevent zeroing of ship_status_info */
    for (index = 0; index < MSG_LENGTH; index++)
        nav_data[index] = ZERO_BYTE;

    // Code to fill in spd position field of nav message
    nav_data[SPD_POS + 3] = (byte)(0x0000000f & ship_status_info[3]);

    // Code to fill in crs position field of nav message
    rem_128 = ship_status_info[2] % 128;
    nav_data[CRS_POS + 3] = (byte)rem_128;
    nav_data[CRS_POS + 2] = (byte)num_128;

    // Code to fill in y position field of nav message
    num_128 = ship_status_info[1] / 128;
    rem_128 = ship_status_info[1] % 128;
    nav_data[Y_POS + 3] = (byte)rem_128;
    nav_data[Y_POS + 2] = (byte)num_128;
    // Code to fill in x position field of nav message
num_128 = ship_status_info[0] / 128;
rem_128 = ship_status_info[0] % 128;
nav_data[X_POS + 3] = (byte)rem_128;
nav_data[X_POS + 2] = (byte)num_128;
nav_data[MSG_TYPE_POS] = NAV_TYPE;
nav_data[REC_TYPE_POS] = NAV_CMD;
nav_data[SUB_REC_POS] = ZERO_BYTE;
nav_data[SER_REC_POS] = ZERO_BYTE;

} // End of build_status_msg

} // End of class Nav_Message

// Class Net_Connection makes socket connection with server
// and exchanges messages with server.
class Net_Connection implements Runnable, Smart_Constants {

// VARIABLES for class Nav_Message
// i_addr - Internet address of client
// client_sock - Socket used by client for exchanging data with server
// input_data - Stream used for collecting data from server
// output_data - Stream used for outputting data to server
// client_receiver - Thread used to decouple receiving and sending data
// receiver_buffer - Byte buffer used for receiving data from server
// new_data - Variable used to indicate if new data has arrived

InetAddress i_addr;
Socket client_sock;
DataInputStream input_data;
DataOutputStream output_data;
Thread client_receiver;
static byte[] receiver_buffer = new byte[MAX_MESSAGE_LENGTH];
static boolean new_data = false;

// Method Net_Connection is constructor for this class
public Net_Connection(InetAddress i_addr) {
this.i_addr = i_addr;
} // End of Net_Connection

// Method send_msg sends messages to server
public boolean send_msg(byte[] msg_to_server) {
if (client_sock == null) {
try {
client_sock = new Socket(i_addr,SERVER_PORT);
input_data = new DataInputStream(client_sock.getInputStream());
output_data = new DataOutputStream(client_sock.getOutputStream());
} catch (Exception e) {
System.out.println("Could not create Socket");
return false;
}
return true;
}
System.out.println(e);
} // End of try - catch logic
} // End of if null
try
{
    output_data.write(msg_to_server);
}
catch (Exception e)
{
    System.out.println("Error writing to socket");
    System.out.println(e);
    return false;
} // End of try - catch block
return true;
} // End of send_msg

// Method run executes thread
public void run()
{
    try
    {
        while (true)
        {
            input_data.read(receiver_buffer); //Client receives data here
            new_data = true;
        } // End of while
    }
catch (Exception e)
{
    System.out.println("Could not receive packet");
    System.out.println(e);
    client_receiver = null;
    return;
} // End of try - catch block
} // End of run

// Method stop discontinues running of thread
public void stop()
{
    if (client_receiver != null)
    {
        client_receiver.stop();
    }
    try
    {
        input_data.close();
        output_data.close();
        client_sock.close();
    }
catch (IOException e)
{
    System.out.println("error closing socket");
    System.out.println(e);
}
} // End of stop

// Method data_receive determines if new data from server has arrived
public boolean data_receive()
{
    return new_data;
} // End of data_receive

// Method clear_data_receive clears new_data variable
public void clear_data_receive()
{
    new_data = false;
} // End of clear_data_receive

// Method get_message retrieves data sent by server for parsing
public void get_message(byte[] received_data)
{
    int index;
    for (index = 0; index < MAX_MESSAGE_LENGTH; index++)
        received_data[index] = receiver_buffer[index];
} // End of get_message
} // End of class Net_Connection
APPENDIX J. RT_SERVER.JAVA FOR TCP

// The sever portion of the RT_RCUUSC software is made up of four Java classes.
// RT_Sever is the driver of the server portion of RT_RCUUSC. This class is
// responsible for instantiation of the Model, Net_S_Connection and Nav_Message
// classes. RT_Server also makes periodical calls to the Model class for updates of the
// model's attributes.

import java.awt.*;
import java.net.*;
import java.io.*;
import java.lang.*;

public class RT_Server
{

    public static void main (String args[])
    {
        new RT_RCUUSC_Server();
    } // End of main

} // End of RT_Server

// RT_RCUUSC_Server is where all supporting objects for the server
// Object are instantiated.
class RT_RCUUSC_Server implements Smart_Constants
{
    Net_S_Connection server_con;
    Nav_Message nav_msg;
    Model rt_model;
    static boolean keep_looping = true;
    static boolean start_model = false;
    final static int INIT_FIELDs = 4;    // Number of fields in init message
    final static int STATUS_FIELDs = 4;  // Number of fields in status fields
    final static int CMD_FIELDs = 3;     // Number of fields in command message
    final static int TENTH_SEC = 100;    // Tenth of sec counter
    final static int TWO_SEC = 21;      // Twenty tenth = 2 secs
    int two_sec_counter = 0;
    static byte[] rec_buffer = new byte[MAX_MESSAGE_LENGTH]; // Byte buffer for receive message
    static byte[] send_buffer = new byte[MAX_MESSAGE_LENGTH]; // Byte buffer for sending message
    static int[] rt_ruccus_init = new int[INIT_FIELDs];      // Integer buffer for initial info
    static int[] rt_ruccus_cmd = new int[CMD_FIELDs];        // Integer buffer for command info
    static int[] rt_ruccus_status = new int[STATUS_FIELDs];  // Integer buffer for status info
    byte rec_type,sub_rec_type;
    FileOutputStream fsnt3_out;    // Output stream for command received from client
    DataOutputStream dsnt3_out;    // Provides methods for writing primitive types
    FileOutputStream fsnt_out;    // Output stream for ship's position and speed
    DataOutputStream dsnt_out;    // Provides methods for writing primitive types
    String ser_num,crs_string,spd_string,time;

    RT_RCUUSC_Server()
```java
try {
    long current_time;
    FileOutputStream fsnt3_out = new FileOutputStream("rec_cmd.txt");
    DataOutputStream dsnt3_out = new DataOutputStream(fsnt3_out);
    FileOutputStream fsnt_out = new FileOutputStream("sent_u.txt");
    DataOutputStream dsnt_out = new DataOutputStream(fsnt_out);
    server_con = new Net_S_Connection(); // Instantiate "network logic class"
    nav_msg = new Nav_Message(); // Instantiate "Navigation logic class"

    // Loops until a stop message from client is received
    while (keep_looping) {
        if (server_con.data_receive()) // Checks if client has sent message
            {
                server_con.clear_data_receive();
                server_con.get_message(rec_buffer);
                rec_type = rec_buffer[MSG_TYPE_POS];
                switch (rec_type) {
                    case NAV_TYPE:
                        
                        sub_rec_type = rec_buffer[REC_TYPE_POS];
                        switch (sub_rec_type) {
                            case NAV_INIT: // Starts ship's position, heading and speed model
                                if (start_model == false) {
                                    try {
                                        start_model = true;
                                        nav_msg.get_ship_init(rec_buffer, rt_ruccus_init);
                                        rt_model = new Model(rt_ruccus_init[0], rt_ruccus_init[1], rt_ruccus_init[2], rt_ruccus_init[3]);
                                        ser_num = Byte.toString(rec_buffer[SER_NUM_LOC]);
                                        dsnt3_out.writeBytes(ser_num);
                                        dsnt3_out.writeBytes(" ");
                                        csr_string = Integer.toString(rt_ruccus_init[2]);
                                        dsnt3_out.writeBytes(csr_string);
                                        spd_string = Integer.toString(rt_ruccus_init[3]);
                                        dsnt3_out.writeBytes(" ");
                                        dsnt3_out.writeBytes(spd_string);
                                        dsnt3_out.writeChar('n'); // output sent messages to file
                                        } // End of try
                                        catch (IOException ioe) {
                                            System.out.println(" Could not write init to rec_cmd.txt output file");
                                        } // End of catch
                                        } // End of start_model = false
                                        break;
                            case NAV_CMD:
                                nav_msg.get_ship_cmd(rec_buffer, rt_ruccus_cmd); // Parse command
                                try
```

102
{  
    ser_num = Byte.toString(rec_buffer[SER_NUM_LOC]);  
    dsnt3_out.writeBytes(ser_num);  
    dsnt3_out.writeBytes(" ");  
    crs_string = Integer.toString(rt_ruccus_cmd[0]);  
    dsnt3_out.writeBytes(crs_string);  
    spd_string = Integer.toString(rt_ruccus_cmd[1]);  
    dsnt3_out.writeBytes(" ");  
    dsnt3_out.writeBytes(spd_string);  
    dsnt3_out.writeChar('n'); // Output command message received from client  
}  
// End of try  
catch (IOException ioe)  
{  
    System.out.println(" Could not write cmd to rec_cmd.txt output file");  
}  
// End of catch  
rt_model.set_cmd_data(rt_ruccus_cmd);  
break;  
default:// Do nothing with nav stat messages  
}  
// End of switch for sub type  
}  
//End of case NAV_TYPE  
break;  
case MODEL_STOP:  
    start_model = false;  
    keep_looping = false;  
    server_con.stop(); // Stop server_con  
try  
{  
    fsnt3_out.close();  
    dsnt_out.close();  
}  
catch (IOException ioe)  
{  
    System.out.println("could not close file");  
}  
break;  
default:  
    // Shouldn't be getting her  
}  
//end of switch for type  
}  
// end of if data_receive  
try  
{  
    Thread.sleep(TENTH_SEC); // Tenth of a second delay  
}  
catch (InterruptedException e) {} // Sleep exception  
if (((two_sec_counter % TWO_SEC) == 0) && (start_model == true))  
{  
    two_sec_counter = 1;  
    rt_model.ship_status(rt_ruccus_status);  
    nav_msg.build_status_msg(send_buffer,rt_ruccus_status);  
    ser_num = Byte.toString(send_buffer[SER_NUM_LOC]);  
    current_time = System.currentTimeMillis();  
    time = Long.toString(current_time);  
    dsnt_out.writeBytes(ser_num);  
    dsnt_out.writeBytes(" ");  
    dsnt_out.writeBytes(time);  
}  
// End of try
dsnt_out.writeChar('n');  // Output sent messages to file
server_con.send_msg(send_buffer);  // Send rt_rucus update to client
}
  // End if 2 second update block
else
two_sec_counter++;  
// End of while keep_looping
}  // End of opening files
catch (IOException ioe)
{
  System.out.println("Could not open rec_cmd.txt output file");
}
}  // End of RT_RCUSC_Server constructor
}  // End of RT_RCUSC_Server class

/**
 * Class Model updates SmartShip position based on a 2 second interval.
 */
class Model
{
  // VARIABLES for class Model
  // array_index - Integer variable used as array index
  // ship_x_pos - Double variable, x position of SmartShip
  // ship_y_pos - Double variable, y position of SmartShip
  // ship_crs - Integer variable, SmartShip course
  // ship_spd - Integer variable, SmartShip speed
  // ship_lat - Integer variable, SmartShip latitude
  // ship_lon - Integer variable, SmartShip longitude
  // ship_cmd_crs - Integer variable, SmartShip command course
  // ship_cmd_spd - Integer variable, SmartShip command speed
  // ship_acceleration - Integer variable, Smartship acceleration
  // deg_per_2sec - Constant Integer, maximum number of degs ship can turn
  // - in a position update interval
  // max_delta - Constant integer, sets heading to cmd heading if
  // - within 5 degs
  // full_compass - Constant integer, number of degrees in a compass
  // half_compass - Constant integer, number of degrees in half a compass
  // FEET_MILE - Constant double, number of feet in mile
  // SECS_HOUR - Constant double, number of seconds in a hour
  // SECS_UPDATE - Constant double, number of seconds between update
  // UNITS - Constant double, number of feet each pixel represents

  static int array_index = 0;
  static double ship_x_pos = 0.0;
  static double ship_y_pos = 0.0;
  static int ship_crs = 0;
  static int ship_spd = 0;
  static int ship_lat = 0;
  static int ship_lon = 0;
  static int ship_cmd_crs = 345;
  static int ship_cmd_spd = 20;
  static int ship_acceleration = 1;
  final int deg_per_2sec = 5;
  final int max_delta = 355;
final int full_compass = 360;
final int half_compass = 180;
final double FEET_MILE = 5280.0;
final double SECS_HOUR = 3600.0;
final double SECS_UPDATE = 3.0;
final double UNITS = 40.0;  // 7/13 was 10 increase to expand map

/** Model(int,int,int,int) Constructor for class Model - 4 parameters initial latitude, longitude,
 * @param srt_lat is of type int - starting latitude of Smart Ship.
 * @param srt_lon is of type int - starting longitude of Smart Ship.
 * @param srt_crs is of type int - starting course of Smart Ship at beginning of interval.
 * @param srt_spd is of type int - starting speed of Smart Ship at beginning of interval.
 * @param ship_cmd_spd of type int - command speed of Smart Ship.
 * @param ship_cmd_crs of type int - command course of Smart Ship.
 * course and speed.
 */

Model(int srt_lat, int srt_lon, int srt_crs, int srt_spd)
{
    ship_lat = srt_lat;
    ship_lon = srt_lon;
    ship_crs = srt_crs;
    ship_spd = srt_spd;
    ship_cmd_spd = srt_spd;
    ship_cmd_crs = srt_crs;
}    // End of Model

/** Method ship_status(int[]) updates current SmartShip position, speed and course.
 * @param status array of int current position ,speed and course of Smart Ship.
 */

public boolean ship_status(int status[])
{
    // VARIABLES for method ship_status
    // delta_crs - Integer variable, delta between heading and
    // command heading
    // rad_to_deg - Double variable, conversion factor for degrees to radians
    // crs_rads - Double variable, heading in radians
    // temp_x - Double variable, used for computation
    // temp_y - Double variable, used for computation
    int delta_crs = 0;
    double rad_to_deg = Math.PI / 180.0;
    double crs_rads;
    double temp_x,temp_y;

    if (ship_crs != ship_cmd_crs)
    {

105
delta_crs = Math.abs(ship_crs - ship_cmd_crs);
if ((delta_crs <= deg_per_2sec) || (delta_crs >= max_delta))
    ship_crs = ship_cmd_crs;
else
    { delta_crs = ship_cmd_crs - ship_crs;
        if ((delta_crs > 0) && (delta_crs <= half_compass))
            ship_crs = ship_crs + deg_per_2sec;
        else if ((delta_crs > 0) && (delta_crs > half_compass))
            ship_crs = ship_crs - deg_per_2sec;
        else if ((delta_crs < 0) && (Math.abs(delta_crs) > half_compass))
            ship_crs = ship_crs + deg_per_2sec;
        else
            ship_crs = ship_crs - deg_per_2sec;
        if (ship_crs > full_compass)
            ship_crs = ship_crs - full_compass;
        if (ship_crs < 0)
            ship_crs = ship_crs + full_compass;
    }
    // End of else delta_crs !<= deg_per_2sec
    // End of cmd_crs != ship_crs
if (ship_spd != ship_cmd_spd)
    {
        if (ship_spd > ship_cmd_spd)
            ship_spd = ship_spd - ship_acceration;
        else
            ship_spd = ship_spd + ship_acceration;
    }
    // End of ship_spd != ship_cmd_spd
crs_rads = ((double) ship_crs * rad_to_deg);
temp_x = (((Math.cos(crs_rads) * (double) ship_spd * FEET_MILE) / (SECS_HOUR * SECS_UPDATE)) / UNITS);
temp_y = (((Math.sin(crs_rads) * (double) ship_spd * FEET_MILE) / (SECS_HOUR * SECS_UPDATE)) / UNITS);
ship_x_pos = ship_x_pos + temp_x;
ship_y_pos = ship_y_pos + temp_y;
status[0] = (int) Math.round(ship_x_pos);
status[1] = (int) Math.round(ship_y_pos);
status[2] = ship_crs;
status[3] = ship_spd;
return true;
}
// End of ship_status

/** Method set_cmd_data(int[]) updates model object with new
 * command speed and heading.
 * @param cmd_info array of int containing current command data for Smart Ship.
 */
public boolean set_cmd_data(int cmd_info[])
{
    ship_cmd_spd = cmd_info[1];
    ship_cmd_crs = cmd_info[0];
    return true;
}
    // End of set_cmd_data
}

// End of class model
// Class Net_S_Connection blocks for socket connection with client,
// sends and receives data with the client.
class Net_S_Connection implements Runnable, Smart_Constants
{

// VARIABLES for class Net_S_Connection
// client_addr - URL address for computer running applet
// server_sock - Server socket used to establish socket with client
// sock - Socket used for exchanging data with client
// input_data - Input stream used to receive data from client
// output_data - Output stream used to send data to client
// server_receiver - Thread used to decouple receiving and sending data
// client_port - Port address blocking for client
// current_time - Variable to record current time
// time - String used to output time
// ser_num - Serial number of a message
// receiver_buffer - Byte buffer to hold input and output messages
// new_data - Variable indicating arrival of new data
// ftn2_out - File used to output STAT messages from client
// dsnt2_out - Used to output STAT messages from client

static InetAddress client_addr;
ServerSocket server_sock;
Socket sock;
DataInputStream input_data;
DataOutputStream output_data;
Thread server_receiver;
int client_port;
long current_time;
String time,ser_num;
static byte[] receiver_buffer = new byte[MAX_MESSAGE_LENGTH];
static boolean new_data = false;
FileOutputStream fnt2_out;
DataOutputStream dnt2_out;

// Constructor method for class Net_S_Connection
public Net_S_Connection()
{
    if (server_receiver == null)
    {
        server_receiver = new Thread(this);
        server_receiver.start();
    } // End of server_receiver
} // End of Net_S_Connection

// Task that blocks until client establishes socket with server
// then receives messages from client.
public void run()
{
    try
    {
        server_sock = new ServerSocket(SERVER_PORT);
    }
sock = server_sock.accept();  // Blocks for a connection
}
catch (Exception e)
{
    System.out.println("Could not create Server_sock");
    System.out.println(e);
}
try
{
    input_data = new DataInputStream(sock.getInputStream());
    output_data = new DataOutputStream(sock.getOutputStream());
    FileOutputStream fsnt2_out = new FileOutputStream("stat_u.txt");
    DataOutputStream dsnt2_out = new DataOutputStream(fsnt2_out);
    try
    {
        while (true)
        {
            try
            {
                input_data.read(receiver_buffer);  // server receives data
            }  // End of try
            catch (IOException e)
            {
                System.out.println("Socket could not receive a packet");
                System.out.println(e);
                System.exit(0);
            }  // End of try catch logic
            if (receiver_buffer[REC_SUB_TYPE_POS] != NAV_STAT)
            {
                new_data = true;  // Might have to implement buffer
            }
        }
        // Outputs NAV_STAT message to file
        ser_num = Byte.toString(receiver_buffer[SER_NUM_LOC]);
        current_time = System.currentTimeMillis();
        time = Long.toString(current_time);
        dsnt2_out.writeBytes(ser_num);
        dsnt2_out.writeBytes(" ");
        dsnt2_out.writeBytes(time);
        dsnt2_out.writeChar('n');  // Output sent messages to file
    }  // Else print out info to file
    }  // End of while
}
catch (IOException e)
{
    System.out.println("error opening streams or output files");
    System.out.println(e);
}  // End of catch for file opening
}  // End of open file
catch (IOException ioe)
{
    System.out.println(" Could not open sent_u.txt output file");
}  // End of catch
}  // End of run

108
// Method send_msg sends ship's position and heading
// to client.
public boolean send_msg(byte[] msg_to_server)
{
    try
    {
        output_data.write(msg_to_server);
    } // End of try
    catch (Exception e)
    {
        System.out.println("Error sending data");
        System.out.println(e);
        return false;
    } // End of try - catch block
    return true;
} // End of send_msg

// Method to unplug socket
public void stop()
{
    if (server_receiver != null)
    {
        server_receiver.stop();
    }
    try
    {
        input_data.close();
        output_data.close();
        server_sock.close();
    } // End of try
    catch (IOException e)
    {
        System.out.println("error closing socket");
        System.out.println(e);
    } // End of catch
    try
    {
        fsmt2_out.close();
    } // End of try
    catch (IOException ioe)
    {
        System.out.println("could not close file");
    } // End of catch
} // End of stop

// Method data_receive indicates if new data has been received.
public boolean data_receive()
{
    return new_data;
} // End of data_receive

// Method clear_data clears new data variable
public void clear_data_receive()
{
    new_data = false;
}  // End of clear_data_receive

//Method get_message retreives data received from client for
// processing.
public void get_message(byte[] received_data)
{
    int index;
    for (index = 0; index < MAX_MESSAGE_LENGTH; index++)
        received_data[index] = receiver_buffer[index];
}  // End of get_message

}  // End of class Net_S_Connection

// Class Nav_Message builds and parses nav messages.
class Nav_Message implements Smart_Constants
{
    // VARIABLES for class Nav_Message
    // MSG_LENGTH - Max length for nav message is 36 bytes
    // NUM_OF_SHIP_PARMS - Nav message contains 4 parameters for ship
    // LAT_PARM_POS - Latitude parameter position in byte buffer
    // LON_PARM_POS - Longitude parameter position in byte buffer
    // CRS_PARM_POS - Course parameter position in byte buffer
    // SPD_PARM_POS - Speed parameter position in byte buffer
    // Y_POS - Ship's y position in byte buffer
    // X_POS - Ship's x position in byte buffer
    // CRS_POS - Ship's course position in byte buffer
    // SPD_POS - Ship's speed position in byte buffer
    // C_CRS_POS - Ship's command course position in byte buffer
    // C_SPD_POS - Ship's command speed position in byte buffer
    // C_ACL_POS - Ship's command acceleration in byte buffer
    // PARM_LENGTH - Parameter's length - 4 bytes
    // ZERO_BYTE - Byte of 0's
    // index2 - Used for loop variable
    // ser_number - Serial number of message
    // MAX_SERIAL - Largest serial number after which value is set to 0
    // nav_data - Byte buffer used for storing messages
    // char_digit - Character used to store a digit as a character

    final static int MSG_LENGTH = 36;
    final static int NUM_OF_SHIP_PARMS = 4;  // x pos,y pos,course and speed
    final static int LAT_PARM_POS = 4;
    final static int LON_PARM_POS = 8;
    final static int CRS_PARM_POS = 12;
    final static int SPD_PARM_POS = 16;
    final static int X_POS = 4;
    final static int Y_POS = 8;
    final static int CRS_POS = 12;
    final static int SPD_POS = 16;
    final static int C_CRS_POS = 4;


110

final static int C_SPD_POS = 8;
final static int C_ACL_POS = 12;
final static int PARM_LENGTH = 4;
final static byte ZERO_BYTE = 0;
int index2;
static byte ser_number = 0;
static final byte MAX_SERIAL = 125;
static byte[] nav_data = new byte[MSG_LENGTH];
char char_digit;

/** Nav_Message Constructor for class */
Nav_Message()
{
}
} // Constructor with no arguments

/** Nav_Message(int,int,int,int) Constructor for class - 4 parameters */
* @param srt_lat is starting latitude of Smart Ship.
* @param srt_lon is starting longitude of Smart Ship.
* @param srt_crs is starting course of Smart Ship.
* @param srt_spd is starting speed of Smart Ship.
* /
Nav_Message(int srt_lat, int srt_lon, int srt_crs, int srt_spd)
{
int index,index2;
int num_length,num_128,rem_128;

for (index = 0; index < MSG_LENGTH; index++)
nav_data[index] = ZERO_BYTE; // Zero out message array

// Code to fill in speed field of nav message
nav_data[SPD_PARM_POS + 3] = (byte)(0x00000007f & srt_spd);

// Code to fill in course field of nav message
num_128 = srt_crs / 128;
rem_128 = srt_crs % 128;
nav_data[CRS_PARM_POS + 3] = (byte)rem_128;
nav_data[CRS_PARM_POS + 2] = (byte)num_128;
nav_data[MSG_TYPE_POS] = NAV_TYPE;
nav_data[REC_TYPE_POS] = NAV_INIT;
nav_data[SUB_REC_POS] = ZERO_BYTE;
nav_data[SER_RET_POS] = ZERO_BYTE;
} // End of method Nav_Message

// Method update_msg updates nav parameters for class
void update_msg(int[] cmd_update)
{
int index,index2,num_128,rem_128;
int num_length;

for (index = 0; index < MSG_LENGTH; index++)
nav_data[index] = ZERO_BYTE; // Zero out message array
// Code to fill in acceleration field of nav message
nav_data[C_ACL_POS + 3] = (byte)(0x0000007f & cmd_update[2]);

// Code to fill in speed field on nav message
nav_data[C_SPD_POS + 3] = (byte)(0x0000007f & cmd_update[1]);

// Code to fill in crs field of nav message
num_128 = cmd_update[0] / 128;
rem_128 = cmd_update[0] % 128;
nav_data[C_CRS_POS + 3] = (byte)rem_128;
nav_data[C_CRS_POS + 2] = (byte)num_128;

nav_data[MSG_TYPE_POS] = NAV_TYPE;
nav_data[REC_TYPE_POS] = NAV_CMD;
nav_data[SUB_REC_POS] = ZERO_BYTE;
nav_data[SER_RET_POS] = ZERO_BYTE;

}  // End of update_msg

//Method get_nav_msg retrieves ship's current data
void get_nav_msg(byte[] init_msg)
{
    int index;
    for (index = 0; index < MSG_LENGTH; index++)
        init_msg[index] = nav_data[index];
}  // End of get init_msg

// Method get_ship_init retrieves ship's initial data
void get_ship_init(byte[] rec_buf,int[] ship_init_info)
{
    int index,num_128,rem_128;
    ship_init_info[0] = 0;  // Zero out lat field
    ship_init_info[1] = 0;  // Zero out lon field
    ship_init_info[2] = 0;  // Zero out the course field
    ship_init_info[3] = 0;  // Zero out the speed field

    // Parse out the speed field of nav message
    ship_init_info[3] = rec_buf[SPD_PARM_POS + 3];

    // Parse out the course field of nav message
    num_128 = rec_buf[CRS_PARM_POS + 2];
    num_128 = num_128 * 128;
    rem_128 = rec_buf[CRS_PARM_POS + 3];
    ship_init_info[2] = num_128 + rem_128;
}  // End of get_ship_init

//Method get_ship_cmd retrieves ship's command data
void get_ship_cmd(byte[] rec_buf,int[] ship_cmd_info)
{
    int index,num_128,rem_128;
    ship_cmd_info[0] = 0;
ship_cmd_info[1] = 0;
ship_cmd_info[2] = 0;

// Parse out the acceleration field of nav message
ship_cmd_info[2] = rec_buf[C_ACL_POS + 3];

// Parse out the speed field of nav message
ship_cmd_info[1] = rec_buf[C_CRSP_POS + 3];

// Parse out the course field of nav message
num_128 = rec_buf[C_SPD_POS + 2];
num_128 = num_128 * 128;
rem_128 = rec_buf[C_SPD_POS + 3];
ship_cmd_info[0] = num_128 + rem_128;

} // End of get_ship_cmd

// Method build_status_msg builds status message to be sent to client
void build_status_msg(byte[] nav_data, int[] ship_status_info)
{
    int index;
    byte num_128, rem_128;
    for (index = 0; index < MSG_LENGTH; index++)
        nav_data[index] = ZERO_BYTE; // Zero out message array

    // Code to fill in spd position field of nav message
    nav_data[SPD_POS + 3] = (byte)(0x0000007F & ship_status_info[3]);

    // Code to fill in crs position field of nav message
    num_128 = (byte)(ship_status_info[2] / 128);
    rem_128 = (byte)(ship_status_info[2] % 128);
    nav_data[CRS_POS + 2] = num_128;
    nav_data[CRS_POS + 3] = rem_128;

    // Code to fill in y position field of nav message
    num_128 = (byte)(ship_status_info[1] / 128);
    rem_128 = (byte)(ship_status_info[1] % 128);
    nav_data[Y_POS + 2] = num_128;
    nav_data[Y_POS + 3] = rem_128;

    // Code to fill in x position field of nav message
    num_128 = (byte)(ship_status_info[0] / 128);
    rem_128 = (byte)(ship_status_info[0] % 128);
    nav_data[X_POS + 2] = num_128;
    nav_data[X_POS + 3] = rem_128;

    nav_data[MSG_TYPE_POS] = NAV_TYPE;
    nav_data[REC_TYPE_POS] = NAV_CMD;
    nav_data[SUB_REC_POS] = ZERO_BYTE;
    // Fill in serial number
    nav_data[SER_NUM_LOC] = ser_number;
    ser_number++;
    if (ser_number > MAX_SERIAL)

113
ser_number = 0;

} // End of build_status_msg

} // End of class Nav_Message
public interface Smart_Constants
{
    final static int BITS_IN_BYTE = 8;  // Number of bits in a byte
    final static int BYTES_IN_INT = 4;  // Number of bytes in an integer
    final static int SER_RET_POS = 3;   // Serial number position in buffer
    final static int MSG_TYPE_POS = 0;  // Message type position in buffer
    final static int REC_TYPE_POS = 1;  // Record type position in buffer
    final static int SUB_REC_POS = 2;   // Sub record type position in buffer
    final static int MAX_MESSAGE_LENGTH = 72;  // Maximum bytes in buffer
    final static int MAX_ARRAY = 25;
    final static int SERVER_PORT = 1997;  // Serial port used for sockets
    final static int REC_SUB_TYPE_POS = 1; // Sub record type position in buffer
    final static byte NAV_TYPE = 1;      // NAV_TYPE value
    final static byte NAV_INIT = 1;      // NAV_INIT value
    final static byte NAV_CMD = 2;       // NAV_CMD value
    final static byte NAV_STAT = 3;      // NAV_STAT value
    final static byte SER_NUM_LOC = 3;   // SER_NUM_LOC type position in buffer
    final static byte MODEL_STOP = 99;   // MODEL_STOP value
}  // End of Smart_Constants
APPENDIX L.  RT_RCUSCHPAGE.JAVA FOR UDP

// RT_RCUSCHpage.java is the client portion of the RT_RCUSC software. It is made
// up of seven classes. RT_RCUSCHpage is the driver of the client program and is
// responsible for instantiation of the Navcontrol, Net_Connection. Nav_message
// NavUpdate, Compass and Plot classes.

import java.awt.*;
import java.net.*;
import java.io.*;

// Class RT_RCUSCHpage - Main class of this applet.
// It's responsible for instantiation of the four
// supporting objects - NavControl, NavUpdate, Compass,
// and Plot.
/** RT_RCUSCHpage.java
 * @version 1.1
 * @author Floyd Bailey
 * Implementation for RT_RCUSCHpage main control
 */

/**
 * Constructor for class RT_RCUSCHpage.
 */

public class RT_RCUSCHpage extends java.applet.Applet
implements Runnable, Smart_Constants
{
    // VARIABLES for RT_RCUSCHpage
    // init_nav - NavControl object declaration
    // update_nav - NavUpdate object declaration
    // compass_update - Compass object declaration
    // plot_update - Plot object declaration
    // ship_model - Model object declaration
    // first_time - Boolean variable indicating whether
    // - initial course and speed was entered
    // i - Integer variable used for exception
    // - handler
    // InitThread - Thread object declaration
    // MAX_ARRAY - Integer constant used to determine the
    // - size of array
    // point_history - Array of Point used to be last MAX_ARRAY
    // - x,y position of RT_RCUSC
    // temp_point - Temporary Point variable
    // array_index - Integer variable indicating the number of
    // - Elements in point_history
    // ship_info - Integer array used to update RT_RCUSC
    // - Position parameters
    // cmd_info - Integer array used to update RT_RCUSC
    // - Command parameters
    // ret_value - Boolean variable assign returning condition
    // - of methods returning boolean type
    // panel1 - Instantiation of Java Panel class
NavControl init_nav;
NavUpdate update_nav;
Compass compass_update;
Plot plot_update;
Nav_Message nav_msg; // Class to handle building and parsing nav
Net_Connection net_con; // Class to handle network connection
static boolean first_time = true;

int i = 0;
Thread InitThread;
static Point[] point_history = new Point[MAX_ARRAY];
Point temp_point,temp_point2;
static int[] index = 0;
static int[] ship_info = new int[4];
static int[] cmd_info = new int[3];
static int update_trigger = 0; // Used to trigger update every 2 sec
boolean ret_value;
String host;
static byte[] msg_buffer = new byte[MAX_MESSAGE_LENGTH];
byte[] rec_buffer = new byte[MAX_MESSAGE_LENGTH];
Panel panel1 = new Panel();
byte record_type;

/** Method init() of class RT_RCUSC - This method instantiates objects
 * of classes NavUpdate, Compass, Plot and NavUpdate.
 */
public void init()
{
    host = this.getCodeBase().getHost();
    init_nav = new NavControl(this, "LAT", "LON", "X_POS", "Y_POS", "COURSE", "SPEED",
            0, 0, 0, 0, 0, 0);
    update_nav = new NavUpdate(this, "COURSE", "SPEED", host, 0, 0, 0);
    compass_update = new Compass(this);
    plot_update = new Plot(this);
    setBackground(Color.white);
    setLayout(new GridLayout(2,2,10,10));
    add(init_nav);
    add(update_nav);
    add(compass_update);
    add(plot_update);
    // Get address of applet's home
}
    // End of init

/** Method insets(int,int,int,int) of class RT_RCUSC - This method
 * determines the spaces between the four panels - init_nav,
 * update_nav, compass_update and plot_update.
 */
public Insets insets()
{ return new Insets(10,10,10,10); // End of insets
}

/** Method start() of class RT_RCUSC - This method instantiates thread
 * InitThread and starts it.
 */
public void start()
{
    if (InitThread == null)
    {
        InitThread = new Thread(this);
        InitThread.start();
    } // End of == null
} // End of start

/** Method stop() of class RT_RCUSC - this method stops thread
 * InitThread and de-instantiates it.
 */
public void stop()
{
    if (InitThread != null)
    {
        InitThread.stop(); // Put back in 5/25
        InitThread = null;
        net_con.stop(); // Disconnect connection
    } // End of != null
} // End of stop

/** Method run() of class RT_RCUSC - This method is the main loop of the
 * class. This is where data is exchanged between the five supporting
 * objects - init_nav, update_nav, compass_update, plot_update and
 * ship_model.
 */
public void run()
{
    while (true) // Loop forever
    {
        if (init_nav.running())
        {
            if (first_time) // Get RT_RCUSC initial speed and course
            {
                try
                {
                    ret_value = init_nav.init_info(ship_info);
                    nav_msg = new Nav_Message(ship_info[0],ship_info[1],ship_info[2],ship_info[3]);
                    nav_msg.get_nav_msg(msg_buffer); // Get msg to send
                    net_con = new Net_Connection(InetAddress.getByName(host));
                    net_con.send_msg(msg_buffer); // Send out init position to server
                }
                catch (Exception e)
                {
                    e.printStackTrace();
                }
            }
        }
    }
}
first_time = false;
}

// End of try
catch (Exception e)
{
    System.out.println(e);
    // End of catch
}

// End of first_time
else
    // RT_RCUSC initial data already entered
{
    if (update_trigger >= 23) // TICKLE VALUE - change to another number ex 27 & recompile
        // Get RT_RCUSC command speed, course and acceleration update
        update_trigger = 0;
        cmd_info[0] = 25; // Speed
        cmd_info[1] = cmd_info[1] + 3; // Course
        if (cmd_info[1] > 359)
            cmd_info[1] = 0;
        nav_msg.update_msg(cmd_info); // Nav_msg class builds update msg
        nav_msg.getNav_msg(msg_buffer); // Get msg to send
        net_con.send_msg(msg_buffer); // Send msg to server
    }
    // End of if update_command
    if (net_con.data_receive()) // Has server sent info
    {
        net_con.clear_data_receive();
        net_con.getMessage(rec_buffer);
        record_type = rec_buffer[0];
        switch (record_type) // Parse server info
        {
            case NAV_TYPE :
                nav_msg.get_ship_info(rec_buffer,ship_info);
                rec_buffer[REC_SUB_TYPE_POS] = NAV_STAT;
                net_con.send_msg(rec_buffer); // Send msg back to server for timing
                break;
            default: // Shouldn't be getting here
                // End of switch logic
                ret_value = compass_update.set_compass(ship_info[2]);
        }
        // End of if data_receive
    if (array_index < MAX_ARRAY)
    {
        temp_point = new Point(ship_info[0], ship_info[1]);
        point_history[array_index] = temp_point;
        array_index++;
    }
    // End of if array not full
else
    {
        temp_point2 = point_history[MAX_ARRAY - 1];
        for (int for_index = (MAX_ARRAY - 1); for_index > 0; for_index--)
        {
            temp_point = point_history[for_index - 1];
            point_history[for_index - 1] = temp_point2;
            temp_point2 = temp_point;
        }
        // End of for loop
        temp_point = new Point(ship_info[0], ship_info[1]);
        point_history[MAX_ARRAY - 1] = temp_point;
    }
    // End of else array not full yet

120
ret_value = plot_update.plot_array(point_history,array_index);
repaint();
compass_update.repaint();
plot_update.repaint();
} // End of else
try
{
    Thread.sleep(100); // one tenth of second delay between position updates
    update_trigger++;
}
catch (InterruptedException e) { } // Sleep exception
// End of if running
repaint();
} // End of while (true) loop
// End of run

/** Method update(NavControl) of class RT_RCUSC - TBD */
void update(NavControl Nav_in)
{
    // Future code can go here
    // End of update NavControl
}

/** Method update(NavUpdate) of class RT_RCUSC - TBD */
void update(NavUpdate Nav_in)
{
    // Future code can go here
    // End of update NavUpdate
}

/** Method paint(Graphics) */
public void paint(Graphics g)
{
    g.setColor(Color.red);
    g.drawString(" RUCUS ",0,60);
    // End of pairing
}

/** Class NavControl - Supporting class of RT_RCUSCHpage.*
 * It's responsible for initial RT_RCUSC data
*/
class NavControl extends Panel
{
    // VARIABLES for class NavControl
    // running - Boolean variable which indicates if object is
    //    - running
    // lat - TextField variable to output latitude data to screen
    // lon - TextField variable to output longitude to screen
    // x_pos - TextField variable to output x position to screen
    // y_pos - TextField variable to output y position to screen
    // course - TextField variable to output course to screen
    // speed - TextField variable to output speed to screen
    // r_lat - Integer variable for current latitude of RT_RCUSC
    // r_lon - Integer variable for current longitude of RT_RCUSC
    // r_x_pos - Integer variable for current x position of RT_RCUSC
// r_y_pos - Integer variable for current y position of RT_RCUSe
// r_icourse - Integer variable for current RT_RCUSe heading
// r_ispeed - Integer variable for current RT_RCUSe speed
// MAX_SPEED - Constant maximum speed of RT_RCUSe
// MIN_SPEED - Constant minimum speed of RT_RCUSe
// MAX_CRS - Constant maximum allowable compass input
// MIN_CRS - Constant minimum allowable compass input
// MIN_X - Constant minimum allowable x axis value
// MAX_X - Constant maximum allowable x axis value
// MIN_Y - Constant minimum allowable y axis value
// MAX_Y - Constant maximum allowable y axis value
// MAX_LAT - Constant maximum allowable latitude value
// MIN_LAT - Constant minimum allowable latitude value
// MAX_LON - Constant maximum allowable longitude value
// MIN_LON - Constant minimum allowable longitude value
// start_button - Constant String value for push button label
// stop_button - Constant String value for push button label
// outerparent - Variable used to declare RT_RCUSe page as parent
static boolean running = false;
static TextField lat,lon,x_pos,y_pos,course,speed;
static int r_lat = 0;
static int r_lon = 0;
static int r_x_pos = 0;
static int r_y_pos = 0;
static int r_i course = 0;
static int r_ispeed = 0;
final int MAX_SPD = 45;
final int MIN_SPD = -20;
final int MAX_CRS = 360;
final int MIN_CRS = 0;
final int MIN_X = -1024;
final int MAX_X = 1024;
final int MAX_Y = 1024;
final int MIN_Y = -1024;
final int MAX_LAT = 90;
final int MIN_LAT = -90;
final int MIN_LON = -180;
final int MAX_LON = 180;
final String start_button = "START";
final String stop_button = "STOP";
RT_RCUSe page outerparent;

/** NavControl is the Constructor for class NavControl. It takes 12
   * parameters 6 of type String and 6 parameters of double.
   * @param parm1 string label for latitude of RT_RCUSe.
   * @param parm2 string label for longitude of RT_RCUSe.
   * @param parm3 string label for x position of RT_RCUSe.
   * @param parm4 string label for y position of RT_RCUSe.
   * @param parm5 string label for course of RT_RCUSe.
   * @param parm6 string label for speed of RT_RCUSe.
   * @param flt1 initial Latitude of RT_RCUSe.
   * @param flt2 initial Longitude of RT_RCUSe.
*/
* @param flt3 initial x position of RT_RCUSC 0.
* @param flt4 initial y position of RT_RCUSC 0.
* @param flt5 initial course of RT_RCUSC.
* @param flt6 initial speed of RT_RCUSC.
 */

NavController(RT_RCUSCHpage target, String parm1, String parm2, String parm3,
            String parm4, String parm5, String parm6, double flt1, double flt2,
            double flt3, double flt4, double flt5, double flt6)
{
    this.outparent = target;
    setLayout(new GridLayout(7, 2, 10, 10)); // Sets up a grid of 7 rows and 2 columns

    // Convert float parameters to string
    lat = new TextField(String.valueOf(flt1), 6);
    lon = new TextField(String.valueOf(flt2), 6);
    x_pos = new TextField(String.valueOf(flt3), 6);
    y_pos = new TextField(String.valueOf(flt4), 6);
    course = new TextField(String.valueOf(flt5), 6);
    speed = new TextField(String.valueOf(flt6), 6);

    // Set edit boxes to a white background
    lat.setBackground(Color.white);
    lon.setBackground(Color.white);
    x_pos.setBackground(Color.white);
    y_pos.setBackground(Color.white);
    course.setBackground(Color.white);
    speed.setBackground(Color.white);

    // Create label and text fields
    add(new Label(parm1, Label.RIGHT));
    add(lat);
    add(new Label(parm2, Label.RIGHT));
    add(lon);
    add(new Label(parm3, Label.RIGHT));
    add(x_pos);
    add(new Label(parm4, Label.RIGHT));
    add(y_pos);
    add(new Label(parm5, Label.RIGHT));
    add(course);
    add(new Label(parm6, Label.RIGHT));
    add(speed);

    // Create and label two buttons
    add(new Button("START"));
    add(new Button("STOP"));
    setBackground(Color.green);
} // End of constructor NavController

/** Method insets up the spacing between items in this panel - labels, text boxes
 * and buttons.
 */
public Insets insets()
{
    return new Insets(10,10,10,150);
} // End of insets

/** Method init_info loads initial latitude, longitude, course and
 *  speed into an array.
 */
public boolean init_info(int info[])
{
    info[0] = r_lat;
    info[1] = r_lon;
    info[2] = r_icourse;
    info[3] = r_ispeed;
    return true;
} // End of init_info

/** Method action(Event,Object) handles selection of buttons
 *  and TextFields on this panel.
 *  @param evt type of event to trigger this action
 *  @param arg object type causing this action
 */
public boolean action(Event evt, Object arg)
{
    if (evt.target instanceof TextField)
    {
        outerparent.update(this);
        return true;
    } // End of if true
    else if (evt.target instanceof Button)
    {
        String label = (String)arg;
        if (label.equals(start_button))
        {
            read_values();
            return true;
        } // End of if START button was selected
        else
        {
            running = false;
            return true;
        } // STOP button was selected.
    }
    else
    return false;
} // End of action

124
/** Method running returns a boolean type indicating what 
* state this panel is in - running or not running. 
*/

public boolean running()
{
    return running;  // End of running
}

/** Method read_values reads and checks for values entered by the user. 
*/

public void read_values()
{
    // VARIABLES for method read_values
    // crs_string   - String variable used to store course data
    // spd_string   - String variable used to store speed data
    // xpos_string  - String variable used to store x position data
    // ypos_string  - String variable used to store y position data
    // lat_string   - String variable used to store latitude data
    // lon_string   - String variable used to store longitude data
    // crs_good     - Boolean variable indicating proper course entry
    // spd_good     - Boolean variable indicating proper speed entry
    // xpos_good   - Boolean variable indicating proper x position entry
    // ypos_good   - Boolean variable indicating proper y position entry
    // lat_good     - Boolean variable indicating proper latitude entry
    // lon_good     - Boolean variable indicating proper longitude entry
    // tmp_spd      - Integer variable used for computations
    // tmp_crs      - Integer variable used for computations
    // tmp_xpos     - Integer variable used for computations
    // tmp_ypos     - Integer variable used for computations
    // tmp_lat      - Integer variable used for computations
    // tmp_lon      - Integer variable used for computations

    String crs_string;
    String spd_string;
    String xpos_string;
    String ypos_string;
    String lat_string;
    String lon_string;
    boolean crs_good;
    boolean spd_good;
    boolean xpos_good;
    boolean ypos_good;
    boolean lat_good;
    boolean lon_good;
    int tmp_spd = r_ispeed;
    int tmp_crs = r_icourse;
    int tmp_xpos = r_x_pos;
    int tmp_ypos = r_y_pos;
    int tmp_lat = r_lat;
    int tmp_lon = r_lon;

    // Next 7 instructions assigns values enter by user into String variables
    crs_string = course.getText();
    spd_string = speed.getText();

    125
xpos_string = x_pos.getText();
ypos_string = y_pos.getText();
lat_string = lat.getText();
lon_string = lon.getText();

// Initialize data as valid.
crs_good = spd_good = xpos_good = ypos_good = lat_good = lon_good = true;

// Following try block parses user data and checks for correct range and type.
try {
    tmp_crs = Integer.parseInt(crs_string);
    if ((tmp_crs > MAX_CRS) || (tmp_crs < MIN_CRS))
    {
        crs_good = false;
        course.setBackground(Color.red);
        course.setText(String.valueOf(r_icourse));
    } // End of failed course entry
    tmp_spd = Integer.parseInt(spd_string);
    if ((tmp_spd > MAX_SPD) || (tmp_spd < MIN_SPD))
    {
        spd_good = false;
        speed.setBackground(Color.red);
        speed.setText(String.valueOf(r_ispeed));
    } // End of failed speed entry
    tmp_xpos = Integer.parseInt(xpos_string);
    if ((tmp_xpos > MAX_X) || (tmp_xpos < MIN_X))
    {
        xpos_good = false;
        x_pos.setBackground(Color.red);
        x_pos.setText(String.valueOf(r_x_pos));
    } // End of failed x pos entry
    tmp_ypos = Integer.parseInt(ypos_string);
    if ((tmp_ypos > MAX_Y) || (tmp_ypos < MIN_Y))
    {
        ypos_good = false;
        y_pos.setBackground(Color.red);
        y_pos.setText(String.valueOf(r_y_pos));
    } // End of failed y pos entry
    tmp_lat = Integer.parseInt(lat_string);
    if ((tmp_lat > MAX_LAT) || (tmp_lat < MIN_LAT))
    {
        lat_good = false;
        lat.setBackground(Color.red);
        lat.setText(String.valueOf(r_lat));
    } // End of failed lat entry
    tmp_lon = Integer.parseInt(lon_string);
    if ((tmp_lon > MAX_LON) || (tmp_lon < MIN_LON))
    {
        lon_good = false;
        lon.setBackground(Color.red);
        lon.setText(String.valueOf(r_lon));
    } // End of failed lon entry
} // End of try

126
catch (Exception e) {

} // End of catch
if ((crs_good == true) && (spd_good == true) && (xpos_good == true) &&
   (ypos_good == true) && (lat_good == true) && (lon_good == true))
{
    course.setBackground(Color.white);
    speed.setBackground(Color.white);
    x_pos.setBackground(Color.white);
    y_pos.setBackground(Color.white);
    lat.setBackground(Color.white);
    lon.setBackground(Color.white);
    course.setText(String.valueOf(tmp_crs));
    r_icourse = tmp_crs;
    speed.setText(String.valueOf(tmp_spd));
    r_ispeed = tmp_spd;
    x_pos.setText(String.valueOf(tmp_xpos));
    r_x_pos = tmp_xpos;
    y_pos.setText(String.valueOf(tmp_ypos));
    r_y_pos = tmp_ypos;
    lat.setText(String.valueOf(tmp_lat));
    r_lat = tmp_lat;
    lon.setText(String.valueOf(tmp_lon));
    r_lon = tmp_lon;
    running = true;
} // End of good values submitted
} // End of read_values
} // End of class NavController

/** Class NavUpdate is where command data is enter by the user */
class NavUpdate extends Panel {

    // VARIABLES for class NavUpdate
    // course - String variable used to store course data
    // speed - String variable used to store speed data
    // acceleration - String variable used to store acceleration data
    // update_cmd - Boolean variable indicating if command data was
    // - updated
    // outerparent - Object variable indicating that RT_RCUUpdate is
    // - parent to this class
    // r_spd - Integer variable indicating current command speed
    // r_crs - Integer variable indicating current command course
    // r_acl - Integer variable indicating current command acceleration
    // MAX_SPD - Constant integer, maximum speed for RT_RCUUpdate
    // MIN_SPD - Constant integer, minimum speed for RT_RCUUpdate
    // MAX_CRS - Constant integer, maximum course (degrees) for course
    // MIN_CRS - Constant integer, minimum course (degrees) for course
    // MAX_ACL - Constant integer, maximum acceleration for RT_RCUUpdate
// MIN_ACL - Constant integer, minimum acceleration for RT_RCUSEC
static TextField course, speed, acceleration;
static boolean update_cmd = false;
RT_RCUSECPanel outerparent;
static int r_spd = 0;
static int r_crs = 0;
static int r_acl = 0;
final static int MAX_SPD = 45;
final static int MIN_SPD = -20;
final static int MAX_CRS = 360;
final static int MIN_CRS = 0;
final static int MAX_ACL = 6;
final static int MIN_ACL = -6;

/** Constructor for class NavUpdate - takes 6 parameters. 3 of type String and
 * 3 of type Integer.
 * @param target
 * @param parm1 string type used as a label for Course text field.
 * @param parm2 string type used as a label for Speed text field.
 * @param parm3 string type used as a label for Acceleration text field.
 * @param flt1 integer type used for course enter by user.
 * @param flt2 integer type used for speed enter by user.
 * @param flt3 integer type used for acceleration enter by user.
 */
NavUpdate(RT_RCUSECPanel target, String parm1, String parm2, String parm3,
   int flt1, int flt2, int flt3)
{
   this.outerparent = target;
   setBackground(Color.blue);
   // Next instruction sets up a grid for this panel - 4 rows by 2 columns.
   setLayout(new GridLayout(4,2,10,40));
   String blank = "   "; // Used to align submit button

   // Convert float parameters to string
   course = new TextField(String.valueOf(flt1),5);
   speed = new TextField(String.valueOf(flt2),5);
   acceleration = new TextField(String.valueOf(flt3),5);

   // Set TextField background to white.
   course.setBackground(Color.white);
   speed.setBackground(Color.white);
   acceleration.setBackground(Color.white);

   // Create label and text fields
   add(new Label(parm1, Label.RIGHT));
   add(course);
   add(new Label(parm2, Label.RIGHT));
   add(speed);
   add(new Label(parm3, Label.RIGHT));
   add(acceleration);
   add(new Label(blank, Label.RIGHT));
   add(new Button("SUBMIT"));
} // End of constructor NavUpdate
/** Method insets determines spaces between item in this panel - TextFields, labels
 * and button.
 */
public Insets insets()
{
    return new Insets(10,10,10,70);
} // End of insets

/** This method get_cmd_data(int cmd_info[]) the current command data.
 * @param cmd_info[] is an integer array containing command data
 * The values are read into an array which the
 * caller can use to update RT_RCUSC position.
 */
public boolean get_cmd_data(int cmd_info[])
{
    cmd_info[0] = r_spd;
    cmd_info[1] = r_crs;
    cmd_info[2] = r_acl;
    return true;
} // End of get_cmd_data

/** Method action handles TextField and button action of this panel.
 * @param evt is of type Event, it's the event of this action.
 * @param arg is of type Object, it's the object of this action.
 */
public boolean action(Event evt, Object arg)
{
    if (evt.target instanceof TextField)
    {
        outerparent.update(this);
        return true;
    } // End of if true
    else if (evt.target instanceof Button)
    {
        read_values();
        update_cmd = true;
        return true;
    } // End of else if Button
    else
        return false;
} // End of action

/** Method update_command() returns a boolean indicating if command
 * data has been updated.
 */
public boolean update_command()
{
    return update_cmd;
} // End of update_command

/** Method clear_update_cmd() set update_cmd to false.
 */
public void clear_update_cmd()
```java
{  
    update_cmd = false;
}  
// End of clear_update

/** Method read_values() validates and assigns command data entered by user. */

public void read_values()
{
    // VARIABLES for method read_values()
    // crs_string - String variable used to store course data
    // spd_string - String variable used to store speed data
    // acl_string - String variable used to store acceleration data
    // crs_good - Boolean variable indicating valid course entry
    // spd_good - Boolean variable indicating valid speed entry
    // acl_good - Boolean variable indicating valid acceleration
    // tmp_spd - Integer used for computation
    // tmp_crs - Integer used for computation
    // tmp_acl - Integer used for computation

    String crs_string;
    String spd_string;
    String acl_string;
    boolean crs_good;
    boolean spd_good;
    boolean acl_good;
    int tmp_spd;
    int tmp_crs;
    int tmp_acl;

    // Next 3 instruction assigns command data to String variables.
    crs_string = course.getText();  // Get text of course text box
    spd_string = speed.getText();   // Get text of speed text box
    acl_string = acceleration.getText();  // Get text of acceleration text box

    tmp_spd = r_spd;
    tmp_crs = r_crs;
    tmp_acl = r_acl;

    //Initialize valid data to true.
    crs_good = spd_good = acl_good = true;
    try
    {
        tmp_crs = Integer.parseInt(crs_string);
        if ((tmp_crs > MAX_CRS) || (tmp_crs < MIN_CRS))
        {
            crs_good = false;
            course.setBackground(Color.red);
            course.setText(String.valueOf(r_crs));
        }  
        // End of failed course entry
        tmp_spd = Integer.parseInt(spd_string);
        if ((tmp_spd > MAX_SPD) || (tmp_spd < MIN_SPD))
        {
            spd_good = false;
            speed.setBackground(Color.red);
            speed.setText(String.valueOf(r_spd));
        }
    }
    finally
    {
        // End of failed speed entry
    }
}
```

130
public void NavUpdate()
{
    // End of failed speed entry
    tmp_acl = Integer.parseInt(acl_string);
    if ((tmp_acl > MAX_ACL) || (tmp_acl < MIN_ACL))
    {
        acl_good = false;
        acceleration.setBackgroundColor(Color.red);
        acceleration.setText(String.valueOf(r_acl));
    }
    // End of failed accleration entry
    // End of try
    catch (Exception e)
    {

    }
    // End of catch
    if ((crs_good == true) && (spd_good == true) && (acl_good == true))
    {
        course.setBackgroundColor(Color.white);
        speed.setBackgroundColor(Color.white);
        acceleration.setBackgroundColor(Color.white);
        course.setText(String.valueOf(tmp_crs));
        r_crs = tmp_crs;
        speed.setText(String.valueOf(tmp_spd));
        r_spd = tmp_spd;
        acceleration.setText(String.valueOf(tmp_acl));
        r_acl = tmp_acl;
    }
    // End of good values submitted
    // End of read_values

}
// End of class NavUpdate

/** Class compass displays the current RT_RCUSC heading. */
class Compass extends Canvas
{
    // VARIABLES for class Compass
    // COMPASS_X_CEN - Constant integer compass x center
    // COMPASS_Y_CEN - Constant integer compass y center
    // MAX_Y - Constant integer used for drawing needle
    // MAX_X - Constant integer used for drawing needle
    // MAX_LIT_Y - Constant integer used for drawing needle
    // degree - Double variable, current compass value
    // degs_rads - Double variable, radian value of the heading
    // x_values - Integer array used to draw needle
    // y_values - Integer array used to draw needle
    // pts - Integer variable, number of points to draw needle
    // rad_to_deg - Double variable, conversion factor of degrees to rads
    // outparent - Variable indicating RT_RCUSCHpage as parent
    // cos_ret - Double variable, cosine of heading
    // sin_ret - Double variable, sine of heading

    final int COMPASS_X_CEN = 130;
    final int COMPASS_Y_CEN = 110;
    final int MAX_Y = 50;
    final int MAX_LIT_Y = 5;
    final int MAX_X = 10;
static double degree = 30.0;
double degs_rads;
int x_values[] = {140,130,120,130,140};
int y_values[] = {110,60,110,113,110};
int pts = x_values.length;
double rad_to_deg = Math.PI / 180.0;
RT_RCUSCHpage outerparent;
double cos_ret,sin_ret;

/** Constructor Compass(RT_RCUSCHpage) creates an object of class Compass
 * @param target is of type RT_RCUSCHpage
 */
Compass(RT_RCUSCHpage target)
{
    this.outerparent = target;
    setBackground(Color.blue);
}     // End of Compass

/** Method set_compass(int) sets new heading for compass.
 * @param compass_value is of type int, sets compass to new value.
 */
public boolean set_compass(int compass_value)
{
    degree = (double) compass_value;
    return true;
}     // Set_compass

/** Method paint(Graphics) displays current heading.
 * @param g is of type Graphics, use to paint to screen.
 */
public void paint(Graphics g)
{
    degs_rads = degree * rad_to_deg;
    cos_ret = Math.cos(degs_rads);
    sin_ret = Math.sin(degs_rads);
    x_values[0] = (int) (Math.round(cos_ret * MAX_X) + COMPASS_X_CEN);
    y_values[0] = (int) (Math.round(sin_ret * MAX_X) + COMPASS_Y_CEN);
    x_values[1] = (int) (Math.round(sin_ret * MAX_Y) + COMPASS_X_CEN);
    y_values[1] = (int) (COMPASS_Y_CEN - Math.round(cos_ret * MAX_Y));
    x_values[2] = (int) (COMPASS_X_CEN - Math.round(cos_ret * MAX_X));
    y_values[2] = (int) (COMPASS_Y_CEN - Math.round(sin_ret * MAX_X));
    x_values[3] = (int) (COMPASS_X_CEN - Math.round(sin_ret * MAX_LIT_Y));
    y_values[3] = (int) (COMPASS_Y_CEN + Math.round(cos_ret * MAX_LIT_Y));
    x_values[4] = x_values[0];
    y_values[4] = y_values[0];
    // Next instruction instantiates a new polygon object - the needle
    Polygon comp_needle = new Polygon(x_values,y_values,pts);
    g.setColor(Color.white);
    g.fillOval(50,30,160,160);
    g.drawString("360",120,25);
    g.drawString("180",120,205);
    g.drawString("270",31,116);
    g.drawString("90",212,116);
    g.setColor(Color.black);
g.drawLine(130,30,130,190);
g.drawLine(50,110,210,110);
g.fillPolygon(comp_needle);
} // End of paint
} // End of class Compass

/** Class Plot plots past current (at grid center) and up to the last 15
 * RT_RCUSChpage positions.
 */
class Plot extends Canvas
{
    // VARIABLES for class Plot
    // outerparent - Variable that indicates that RT_RCUSChpage is parent
    // MAX_ARRAY - Constant int - maximum number of points that can be plotted
    // ship_history - Array of type points - positions of RT_RCUSCh
    // last_point - Number of positions in ship_history
    // x_center - String variable used to display x value of grid center
    // y_center - String variable used to display y value of grid center

    RT_RCUSChpage outerparent;
    static final int MAX_ARRAY = 25;
    static Point[] ship_history = new Point[MAX_ARRAY];
    static int last_point = 0;
    String x_center, y_center;

    /** Constructor Plot(RT_RCUSChpage) used to create an instance of class Plot.
     * @param target is of type RT_RCUSChpage
     */
    Plot(RT_RCUSChpage target)
    {
        this.outerparent = target;
        setBackground(Color.green);
    } // End of Compass

    /** Method plot_array(Point[], int) updates RT_RCUSC positions array.
     * @param ship_hist[] of type Point array - last positions of RT_RCUSC.
     * @param last_entry of type int - is the number of postions in array.
     */
    public boolean plot_array(Point ship_hist[], int last_entry)
    {
        // VARIABLES for method plot_array
        // index - Integer variable used as array index
        // index2 - Integer variable used as array index
        // keep_looming - Boolean variable used for loop control

        int index;
        int index2;
        boolean keep_looming = true;
        index2 = (last_entry - 1);
        index = 0;
        while (keep_looming)
        {
            ship_history[index] = ship_hist[index2];
if ((index2 <= 0) || (index >= (MAX_ARRAY - 1)))
    keep_looping = false;
else
    {
        index++;
        index2--;
    }  // End of else
}  // End of while keep_looping
last_point = index;
return true;
}  //End of plot_array

/** Method paint(Graphics) plots RT_RCUSC positions to panel. */
public void paint(Graphics g)
{
    // VARIABLES for method paint(Graphics)
    // REC_WIDTH  - Constant integer, grid width
    // REC_LENGTH  - Constant integer, grid length
    // index      - Integer variable used for array index
    // prev_x     - Integer variable used for line computations
    // prev_y     - Integer variable used for line computations
    // cur_x      - Integer variable used for line computations
    // cur_y      - Integer variable used for line computations

    final int REC_WIDTH = 180;
    final int REC_LENGTH = 180;
    int index,prev_x,prev_y,cur_x,cur_y;
    prev_x = 0;
    prev_y = 0;
    g.setColor(Color.white);
    g.fillRect(50,15,REC_WIDTH,REC_LENGTH);
    g.setColor(Color.black);
    g.drawLine(140,15,140,195);
    g.drawLine(50,105,230,105);
    x_center = Integer.toString(ship_history[0].x);
    y_center = Integer.toString(ship_history[0].y);
    g.drawString(x_center,130,12);
    g.drawString(y_center,20,105);
    for (index = 0; index <= last_point; index++)
    {
        if (index > 0)
        {
            if ((Math.abs(ship_history[index].x) < REC_WIDTH) &&
                (Math.abs(ship_history[index].y) < REC_LENGTH))
            {
                cur_x = ship_history[index].x - ship_history[0].x;
                cur_x = -1 * cur_x;
                cur_y = ship_history[index].y - ship_history[0].y;
                g.setColor(Color.green);
                g.drawLine((140 + prev_y),(105 + prev_x),(140 + cur_y),(105 + cur_x));
                g.setColor(Color.black);
                prev_x = cur_x;
                prev_y = cur_y;
            }
// End of boundary check
}
} // End of if > 0
}
} // End of for loop
}
} // End of paint
}
} // End of class Plot

// Class Nav_Message builds and parses nav messages.
class Nav_Message implements Smart_Constants
{
    final static int MSG_LENGTH = 36;
    final static int NUM_OF_SHIP_PARMS = 4; // x pos, y pos, course and speed
    final static int LAT_PARM_POS = 4;
    final static int LON_PARM_POS = 8;
    final static int CRS_PARM_POS = 12;
    final static int SPD_PARM_POS = 16;
    final static int X_POS = 4;
    final static int Y_POS = 8;
    final static int CRS_POS = 12;
    final static int SPD_POS = 16;
    final static int C_CRS_POS = 4;
    final static int C_SPD_POS = 8;
    final static int C_ACL_POS = 12;
    final static int PARM_LENGTH = 4;
    final static byte ZERO_BYTE = 0;

    static byte[] nav_data = new byte[MSG_LENGTH];
    char char_digit;

    /** Nav_Message Constructor for class */
    Nav_Message()
    {
    }

    // End of constructor with no arguments

    /** Nav_Message(int,int,int,int) Constructor for class - 4 parameters
     * @param srt_lat is starting latitude of RT_RCUSC.
     * @param srt_lon is starting longitude of RT_RCUSC.
     * @param srt_crs is starting course of RT_RCUSC.
     * @param srt_spd is starting speed of RT_RCUSC.
     */
    Nav_Message(int srt_lat, int srt_lon, int srt_crs, int srt_spd)
    {
        int index, index2;
        int num_length, num_128, rem_128;
        for (index = 0; index < MSG_LENGTH; index++)
            nav_data[index] = ZERO_BYTE; // Zero out message array

        // Code to fill in speed field of nav message
        nav_data[SPD_PARM_POS + 3] = (byte)(0x0000007f & srt_spd);

        // Code to fill in course field of nav message
num_128 = srt_crs / 128;
nav_data[CRS_PARM_POS + 2] = (byte)num_128;
rem_128 = srt_crs % 128;
nav_data[CRS_PARM_POS + 3] = (byte)rem_128;

nav_data[MSG_TYPE_POS] = NAV_TYPE;
nav_data[REC_TYPE_POS] = NAV_INIT;
nav_data[SUB_REC_POS] = ZERO_BYTE;
nav_data[SER_RET_POS] = ZERO_BYTE;

} // End of method Nav_Message

// Method update_msg updates nav parameters for class
void update_msg(int[] cmd_update)
{
    int num_of_128, remainder;
    int num_length;
    int index;
    for (index = 0; index < MSG_LENGTH; index++)
        nav_data[index] = ZERO_BYTE; // Zero out message array
    nav_data[C_ACL_POS + 3] = (byte)(0x0000007f & cmd_update[2]); // Acceleration
    nav_data[C_SPD_POS + 3] = (byte)(0x0000007f & cmd_update[0]); // Speed
    nav_data[C_CRS_POS + 2] = (byte)num_of_128;
    remainder = cmd_update[1] % 128;
    nav_data[C_CRS_POS + 3] = (byte)remainder;
    nav_data[MSG_TYPE_POS] = NAV_TYPE;
    nav_data[REC_TYPE_POS] = NAV_CMD;
    nav_data[SUB_REC_POS] = ZERO_BYTE;
    nav_data[SER_RET_POS] = ZERO_BYTE;

} // End of update_msg

// Method get_nav_msg retrieves nav message
void get_nav_msg(byte[] init_msg)
{
    int index;
    for (index = 0; index < MSG_LENGTH; index++)
        init_msg[index] = nav_data[index];
} // End of get init_msg

// Method get_ship_info retrieves ship's position and course
void get_ship_info(byte[] rec_buf, int[] ship_info)
{
    int index, num_128, rem_128;

    // Code to fill in spd field of nav message
    ship_info[3] = 0;
    ship_info[3] = rec_buf[SPD_POS + 3];

    // Code to fill in crs field of nav message
    ship_info[2] = 0;
    num_128 = rec_buf[CRS_POS + 2];
    num_128 = num_128 * 128;

    // Other code...

136
rem_128 = rec_buf[CRS_POS + 3];

// Code to fill in y pos field of nav message
ship_info[1] = 0;
num_128 = rec_buf[Y_POS + 2];
num_128 = num_128 * 128;
rem_128 = rec_buf[Y_POS + 3];
ship_info[1] = num_128 + rem_128;

// Code to fill in x pos field of nav message
ship_info[0] = 0;
num_128 = rec_buf[X_POS + 2];
num_128 = num_128 * 128;
rem_128 = rec_buf[X_POS + 3];
ship_info[0] = num_128 + rem_128;

} // End of get_ship_info

// Method get_ship_init retrieves ship's initialization info
void get_ship_init(byte[] rec_buf, int[] ship_init_info)
{
int index, num_128, rem_128;
ship_init_info[0] = 0; // Zero out lat field
ship_init_info[1] = 0; // Zero out lon field
ship_init_info[2] = 0; // Zero out the course field
ship_init_info[3] = 0; // Zero out the speed field

// Parse out the speed field of nav message
ship_init_info[3] = rec_buf[SPD_PARM_POS + 3];

// Parse out the course field of nav message
num_128 = rec_buf[CRS_PARM_POS + 2];
num_128 = num_128 * 128;
rem_128 = rec_buf[CRS_PARM_POS + 3];
ship_init_info[2] = num_128 + rem_128;

} // End of get_ship_init

// Method get_ship_cmd retrieves ship's command course, speed and acceleration
void get_ship_cmd(byte[] rec_buf, int[] ship_cmd_info)
{
int index, num_128, rem_128;
ship_cmd_info[0] = 0;
ship_cmd_info[1] = 0;
ship_cmd_info[2] = 0;

// Parse out the acceleration field of nav message
ship_cmd_info[2] = rec_buf[C_ACL_POS + 3];

// Parse out the speed field of nav message
ship_cmd_info[1] = rec_buf[C_SPD_POS + 3];

// Parse out the course field of nav message
num_128 = rec_buf[C_CRS_POS + 2];
num_128 = num_128 * 128;
rem_128 = rec_buf[C_CRS_POS + 3];
ship_cmd_info[0] = num_128 + rem_128;

} // End of get_ship_cmd

// Method build_status_msg parses status message received from server
void build_status_msg(byte[] nav_data, int[] ship_status_info)
{
    int index, num_128, rem_128;
    int temp_value; // Used to prevent zeroing of ship_status_info
    for (index = 0; index < MSG_LENGTH; index++)
    {
        nav_data[index] = ZERO_BYTE; // Zero out message array
    }

    // Code to fill in spd position field of nav message
    nav_data[SPD_POS + 3] = (byte)(0x0000007f & ship_status_info[3]);

    // Code to fill in crs position field of nav message
    rem_128 = ship_status_info[2] % 128;
    nav_data[CRS_POS + 3] = (byte)rem_128;
    nav_data[CRS_POS + 2] = (byte)num_128;

    // Code to fill in y position field of nav message
    num_128 = ship_status_info[1] / 128;
    rem_128 = ship_status_info[1] % 128;
    nav_data[Y_POS + 3] = (byte)rem_128;
    nav_data[Y_POS + 2] = (byte)num_128;

    // Code to fill in x position field of nav message
    num_128 = ship_status_info[0] / 128;
    rem_128 = ship_status_info[0] % 128;
    nav_data[X_POS + 3] = (byte)rem_128;
    nav_data[X_POS + 2] = (byte)num_128;
    nav_data[MSG_TYPE_POS] = NAV_TYPE;
    nav_data[REC_TYPE_POS] = NAV_Cmd;
    nav_data[SUB_REC_POS] = ZERO_BYTE;
    nav_data[SER_RECV_POS] = ZERO_BYTE;

} // End of build_status_msg

} // End of class Nav_Message

// Class Net_Connection makes socket connection with server
// and exchanges messages with server.
class Net_Connection implements Runnable, Smart_Constants // Thread
{
    // VARIABLES for class Nav_Message
    // i_addr - Internet address of client
    // client_sock - Socket used by client for exchanging data with server
    // client_receiver - Thread used to decouple receiving and sending data

}
// receiver_buffer - Byte buffer used for receiving data from server
// new_data - Variable used to indicate if new data has arrived
static InetAddress i_addr;
DatagramSocket client_sock;
Thread client_receiver;
static byte[] receiver_buffer = new byte[MAX_MESSAGE_LENGTH];
static boolean new_data = false;

public Net_Connection(InetAddress i_addr)
{
    this.i_addr = i_addr;
} // End of Net_Connection

public boolean send_msg(byte[] msg_to_server)
{
    if (client_sock == null)
    {
        try
        {
            client_sock = new DatagramSocket();
        }
        catch (Exception e)
        {
            System.out.println("Could not create DatagramSocket");
            System.out.println(e);
        } // End of try - catch logic
    } // End of if null
    DatagramPacket c_packet = new DatagramPacket(msg_to_server, msg_to_server.length, i_addr, SERVER_PORT);
    try
    {
        client_sock.send(c_packet);
    }
    catch (Exception e)
    {
        System.out.println("Error sending packet");
        System.out.println(e);
        return false;
    } // End of try - catch block
    return true;
} // End of send_msg

public void run()
{
    DatagramPacket rec_packet = new DatagramPacket(receiver_buffer, receiver_buffer.length);
    try
    {
        while (true)
        {
            client_sock.receive(rec_packet); // Client receives data here
            new_data = true; // Might have to implement buffer
        } // End of while
    }
    catch (Exception e)
    {
    } // End of try
139


```java
{
    System.out.println("Could not receive packet");
    System.out.println(e);
    client_receiver = null;
    return;
} // End of try - catch block
} // End of run

public void stop()
{
    if (client_receiver != null)
    {
        client_receiver.stop();
    }
    client_sock.close(); // End of stop

public boolean data_receive()
{
    return new_data;
} // End of data_receive

public void clear_data_receive()
{
    new_data = false;
} // End of clear_data_receive

public void get_message(byte[] received_data)
{
    int index;
    for (index = 0; index < MAX_MESSAGE_LENGTH; index++)
    {
        received_data[index] = receiver_buffer[index];
    } // End of get_message

} // End of class Net_Connection
```
APPENDIX M. RT_SERVER.JAVA FOR UDP

// The sever portion of the RT_RCUSC software is made up of four Java classes.
// RT_Sever is the driver of the server position of RT_RCUSC. This class is
// responsible for instantiation of the Model, Net_S_Connection and Nav_Message
// classes. RT_Server also makes periodical calls to the Model class for updates of the
// model's attributes.
import java.awt.*;
import java.net.*;
import java.io.*;
import java.lang.*;

public class RT_Server
{
    public static void main (String args[])
    {
        new RT_RCUSC_Server();
    } // End of main
} // End of RT_Server

class RT_RCUSC_Server implements Smart_Constants
{
    Net_S_Connection server_con;
    Nav_Message nav_msg;
    Model rt_model;
    static boolean keep_looping = true;
    static boolean start_model = false;
    final static int INIT_FIELDS = 4;  // Number of fields in init message
    final static int STATUS_FIELDS = 4;  // Number of fields in status fields
    final static int CMD_FIELDS = 3;  // Number of fields in command message
    final static int TENTH_SEC = 100;  // Tenth of sec counter
    final static int TWO_SEC = 21;  // Twenty tenth = 2 secs
    int two_sec_counter = 0;
    static byte[] rec_buffer = new byte[MAX_MESSAGE_LENGTH];  // Byte buffer for receive message
    static byte[] send_buffer = new byte[MAX_MESSAGE_LENGTH];  // Byte buffer for sending message
    static int[] rt_rcusc_init = new int[INIT_FIELDS];  // Integer buffer for initial info
    static int[] rt_rcusc_cmd = new int[CMD_FIELDS];  // Integer buffer for command info
    static int[] rt_rcusc_status = new int[STATUS_FIELDS];  // Integer buffer for status info
    byte rec_type,sub_rec_type;
    FileOutputStream fsnt3_out;
    DataOutputStream dnnt3_out;
    FileOutputStream fdnt3_out;
    DataOutputStream dsnt_out;
    String ser_num,crs_string,spd_string,time;

    RT_RCUSC_Server()
    {
        try
        {
            long current_time;
            FileOutputStream fsnt3_out = new FileOutputStream("rec_cmd.txt");
        }

    }

    // Provides methods for writing primitive types
    // Provides methods for writing primitive types
DataOutputStream dsnt3_out = new DataOutputStream(fsnt3_out);
FileOutputStream fsnt_out = new FileOutputStream("sent_u.txt");
DataOutputStream dsnt_out = new DataOutputStream(fsnt_out);
server_con = new Net_S_Connection(); // Instantiate "network logic class"
nav_msg = new Nav_Message(); // Instantiate "Navigation logic class"

// Loops until a stop message from client is received
while (keep_looping)
{
    if (server_con.data_receive()) // Checks if client has sent message
    {
        server_con.clear_data_receive();
        server_con.get_message(rec_buffer);
        rec_type = rec_buffer[MSG_TYPE_POS];
        switch (rec_type)
        {
            case NAV_TYPE:
                {
                    sub_rec_type = rec_buffer[REC_TYPE_POS];
                    switch (sub_rec_type)
                    {
                        case NAV_INIT:
                            if (start_model == false)
                            {
                                try
                                {
                                    start_model = true;
                                    nav_msg.get_ship_init(rec_buffer, rt_rcusc_init);
                                    rt_model = new Model(rt_rcusc_init[0], rt_rcusc_init[1], rt_rcusc_init[2], rt_rcusc_init[3]);
                                    ser_num = Byte.toString(rec_buffer[SER_NUM_LOC]);
                                    dsnt3_out.writeBytes(ser_num);
                                    dsnt3_out.writeBytes(" ");
                                    crs_string = Integer.toString(rt_rcusc_init[2]);
                                    dsnt3_out.writeBytes(crs_string);
                                    spd_string = Integer.toString(rt_rcusc_init[3]);
                                    dsnt3_out.writeBytes(" ");
                                    dsnt3_out.writeBytes(spd_string);
                                    dsnt3_out.writeChar("n"); // Output sent messages to file
                                } // End of try
                                catch (IOException ioe)
                                {
                                    System.out.println(" Could not write init to rec_cmd.txt output file");
                                }
                                } // End of start_model = false
                                break;
                        case NAV_CMD:
                            nav_msg.get_ship_cmd(rec_buffer, rt_rcusc_cmd); // Parse command
                            try
                            {
                                ser_num = Byte.toString(rec_buffer[SER_NUM_LOC]);
                                dsnt3_out.writeBytes(ser_num);
                                dsnt3_out.writeBytes(" ");
                            }
crs_string = Integer.toString(rt_rcusc_cmd[0]);
dsn3_out.writeBytes(crs_string);
spd_string = Integer.toString(rt_rcusc_cmd[1]);
dsn3_out.writeBytes(" ");
dsn3_out.writeBytes(spd_string);
dsn3_out.writeChar('n'); // Output command message received from client
} // End of try
catch (IOException ioe)
{
    System.out.println("Could not write cmd to rec_cmd.txt output file");
} // End of catch
rt_model.set_cmd_data(rt_rcusc_cmd);
break;
default: // Do nothing with nav stat messages
} // End of switch for sub type
} // End of case NAV_TYPE
break;
case MODEL_STOP:
    start_model = false;
    keep_looking = false;
    server_con.stop(); // Stop server_con
try
{
    fsnt3_out.close();
    dsnt_out.close();
}
catch (IOException ioe)
{
    System.out.println("could not close file");
}
break;
default: // Shouldn't be getting here
} // End of switch for type
} // End of if data_receive
try
{
    Thread.sleep(TENTH_SEC); // Tenth of a second delay
}
catch (InterruptedException e) {} // Sleep exception
if (((two_sec_counter % TWO_SEC) == 0) && (start_model == true))
{
    two_sec_counter = 1;
    rt_model.ship_status(rt_rcusc_status);
    nav_msg.build_status_msg(send_buffer, rt_rcusc_status);
    ser_num = Byte.toString(send_buffer[SER_NUM_LOC]);
    current_time = System.currentTimeMillis();
    time = Long.toString(current_time);
    dsnt_out.writeBytes(ser_num);
    dsnt_out.writeBytes(" ");
    dsnt_out.writeBytes(time);
    dsnt_out.writeChar('n'); // Output sent messages to file
    server_con.send_msg(send_buffer); // Send rt_rcusc update to client
} // End if
else
```java
    two_sec_counter++;                   // End of while keep_loopying
    }
    }                                      // End of opening files
    catch (IOException ioe)
    {                                          // End of RT_RCUSC_Server constructure
        System.out.println(" Could not open rec_cmd.txt output file");
    }
    }

/** Class Model updates SmartShip position based on a 3 second interval. */
class Model
{
    // VARIABLES for class Model
    // array_index - Integer variable used as array index
    // ship_x_pos - Double variable, x position of SmartShip
    // ship_y_pos - Double variable, y position of SmartShip
    // ship_crs - Integer variable, SmartShip course
    // ship_spd - Integer variable, SmartShip speed
    // ship_lat - Integer variable, SmartShip latitude
    // ship_lon - Integer variable, SmartShip longitude
    // ship_cmd_crs - Integer variable, SmartShip command course
    // ship_cmd_spd - Integer variable, SmartShip command speed
    // ship_accration - Integer variable, Smartship acceleration
    // deg_per_2sec - Constant Integer, maximum number of degs ship can turn
    // max_delta - Constant integer, sets heading to cmd heading if
    // full_compass - Constant integer, number of degrees in a compass
    // half_compass - Constant integer, number of degrees in half a compass
    // FEET_MILE - Constant double, number of feet in mile
    // SECS_HOUR - Constant double, number of seconds in a hour
    // SECS_UPDATE - Constant double, number of seconds between update
    // UNITS - Constant double, number of feet each pixel represents

    static int array_index = 0;
    static double ship_x_pos = 0.0;
    static double ship_y_pos = 0.0;
    static int ship_crs = 0;
    static int ship_spd = 0;
    static int ship_lat = 0;
    static int ship_lon = 0;
    static int ship_cmd_crs = 345;
    static int ship_cmd_spd = 20;
    static int ship_accration = 1;
    final int deg_per_2sec = 5;
    final int max_delta = 355;
    final int full_compass = 360;
    final int half_compass = 180;
    final double FEET_MILE = 5280.0;
    final double SECS_HOUR = 3600.0;
```

144
final double SECS_UPDATE = 3.0;
final double UNITS = 50.0;

/** Model(int,int,int,int) Constructor for class Model - 4 parameters initial latitude, longitude,
 * @param srt_lat is of type int - starting latitude of Smart Ship.
 * @param srt_lon is of type int - starting longitude of Smart Ship.
 * @param srt_crs is of type int - starting course of Smart Ship at begining of interval.
 * @param srt_spd is of type int - starting speed of Smart Ship at begining of interval.
 * @param ship_cmd_spd of type int - command speed of Smart Ship.
 * @param ship_cmd_crs of type int - command course of Smart Ship.
 * course and speed.
 */

Model(int srt_lat, int srt_lon, int srt_crs, int srt_spd)
{
    ship_lat = srt_lat;
    ship_lon = srt_lon;
    ship_crs = srt_crs;
    ship_spd = srt_spd;
    ship_cmd_spd = srt_spd;
    ship_cmd_crs = srt_crs;
} // End of Model

/** Method ship_status(int[]) updates current SmartShip position, speed and course.
 * @param status array of int current position ,speed and course of Smart Ship.
 */

public boolean ship_status(int status[])
{
    // VARIABLES for method ship_status
    // delta_crs - Integer variable, delta between heading and
    //   command heading
    // rad_to_deg - Double variable, conversion factor for degrees to radians
    // crs_rads - Double variable, heading in radians
    // temp_x - Double variable, used for computation
    // temp_y - Double variable, used for computation
    int delta_crs = 0;
    double rad_to_deg = Math.PI / 180.0;
    double crs_rads;
    double temp_x,temp_y;

    if (ship_crs != ship_cmd_crs)
    {
        delta_crs = Math.abs(ship_crs - ship_cmd_crs);
        if ((delta_crs <= deg_per_2sec) || (delta_crs >= max_delta))
            ship_crs = ship_cmd_crs;
        else
            // Further processing...
    }

    // Further processing...

    return true; // Success
}
{ 
    delta_crs = ship_cmd_crs - ship_crs;
    if ((delta_crs > 0) && (delta_crs <= half_compass))
        ship_crs = ship_crs + deg_per_2sec;
    else if ((delta_crs > 0) && (delta_crs > half_compass))  //5/6
        ship_crs = ship_crs - deg_per_2sec;
    else if ((delta_crs < 0) && (Math.abs(delta_crs) > half_compass))
        ship_crs = ship_crs + deg_per_2sec;
    else
        ship_crs = ship_crs - deg_per_2sec;
    if (ship_crs > full_compass)
        ship_crs = ship_crs - full_compass;
    if (ship_crs < 0)
        ship_crs = ship_crs + full_compass;
}  // End of else delta_crs <= deg_per_2sec
}  // End of cmd_crs != ship_crs
if (ship_spd != ship_cmd_spd)
{
    if (ship_spd > ship_cmd_spd)
        ship_spd = ship_spd - ship_acceleration;
    else
        ship_spd = ship_spd + ship_acceleration;
}  // End of ship_spd != ship_cmd_spd

crs_rads = ((double) ship_crs * rad_to_deg);
temp_x = ((Math.cos(crs_rads) * (double) ship_spd * FEET_MILE) /
    (SECS_HOUR * SECS_UPDATE)) / UNITS;
temp_y = ((Math.sin(crs_rads) * (double) ship_spd * FEET_MILE) /
    (SECS_HOUR * SECS_UPDATE)) / UNITS;
ship_x_pos = ship_x_pos + temp_x;
ship_y_pos = ship_y_pos + temp_y;
status[0] = (int) Math.round(ship_x_pos);
status[1] = (int) Math.round(ship_y_pos);
status[2] = ship_crs;
status[3] = ship_spd;
return true;
}  // End of ship_status

/** Method set_cmd_data(int[]) updates model object with new
 * command speed and heading.
 * @param cmd_info array of int containing current command data for Smart Ship.
 */
public boolean set_cmd_data(int cmd_info[])
{
    ship_cmd_spd = cmd_info[1];
    ship_cmd_crs = cmd_info[0];
    return true;
}  // End of set_cmd_data
}  // End of class model

// Class Net_S_Connection blocks for socket connection with client,
// sends and receives data with the client.
class Net_S_Connection implements Runnable, Smart_Constants


{ // VARIABLES for class Net_S_Connection
  // client_addr  - URL address for computer running applet
  // server_sock - Server socket used to establish socket with client
  // server_receiver - Thread for decoupling receiving and sending data
  // input_data - Input stream used to receive data from client
  // output_data - Output stream used to send data to client
  // server_receiver - Thread used to decouple receiving and sending data
  // client_port  - Port address blocking for client
  // ser_num  - Serial number of a message
  // receiver_buffer - Byte buffer to hold input and output messages
  // new_data - Variable indicating arrival of new data
  // fsnt2_out  - File used to output STAT messages from client
  // dsnt2_out - Used to output STAT messages from client

  static InetAddress client_addr;
  DatagramSocket server_sock;
  Thread server_receiver;
  int client_port;
  long current_time;
  String time,ser_num;
  static byte[] receiver_buffer = new byte[MAX_MESSAGE_LENGTH];
  static boolean new_data = false;
  FileOutputStream fsnt2_out;
  DataOutputStream dsnt2_out;

  // Constructor method for class Net_S_Connection
  public Net_S_Connection()
  {
    if (server_receiver == null)
    {
      server_receiver = new Thread(this);
      server_receiver.start();
    } // End of server_receiver
  } // End of Net_S_Connection

  // Task that blocks until client establishes socket with server
  // then receives messages from client.
  public void run()
  {
    try
    {
      server_sock = new DatagramSocket(SERVER_PORT);
    } // End of try
    catch (Exception e)
    {
      System.out.println("Could not create DatagramSocket");
      System.out.println(e);
      System.exit(0);
    } // End of catch
    try
    {
      FileOutputStream fsnt2_out = new FileOutputStream("stat_u.txt");
      147
DataOutputStream dsnt2_out = new DataOutputStream(fsnt2_out);
DatagramPacket rec_packet = new DatagramPacket(receiver_buffer,receiver_buffer.length);
try
{
    while (true)
    {
    try
    {
        server_sock.receive(rec_packet);  // Receive data from client
    }  // End of try
    catch(IOException e)
    {
        System.out.println("DatagramSocket could not receive a packet");
        System.out.println(e);
        System.exit(0);
    }  // End of try catch logic
    if (receiver_buffer[REC_SUB_TYPE_POS] != NAV_STAT)
    {
        new_data = true;
        client_addr = rec_packet.getAddress();
        client_port = rec_packet.getPort();
    }  // End of if not NAV_STAT
    else
    {
        ser_num = Byte.toString(receiver_buffer[SER_NUM_LOC]);
        current_time = System.currentTimeMillis();
        time = Long.toString(current_time);
        dsnt2_out.writeBytes(ser_num);
        dsnt2_out.writeBytes(" ");
        dsnt2_out.writeBytes(time);
        dsnt2_out.writeChar('n');  // Output sent messages to file
    }  // Else print out info to file
    }  // End of while
}  // End of try

catch (Exception e)
{
    System.out.println("Could not receive packet");
    System.out.println(e);
    server_receiver = null;
    return;
}  // End of try - catch block
}  // End of open file
catch (IOException ioe)
{
    System.out.println("Could not open sent_u.txt output file");
}  // End of catch
}  // End of run

// Method send_msg sends ship's position and heading
// to client.
public boolean send_msg(byte[] msg_to_server)
{
    DatagramPacket c_packet = new
DatagramPacket(msg_to_server,msg_to_server.length,client_addr,client_port);
try
{
    server_sock.send(c_packet);
}  // End of try
catch (Exception e)
{
    System.out.println("Error sending packet");
    System.out.println(e);
    return false;
}  // End of try - catch block
return true;  // End of send_msg

// Method to unplug socket
public void stop()
{
    if (server_receiver != null)
    {
        server_receiver.stop();  // End of if not null
    }
    server_sock.close();
    try
    {
        fsnt2_out.close();
    }  // End of try
    catch (IOException ioe)
    {
        System.out.println("could not close file");
    }  // End of catch
}  // End of stop

// Method data_receive indicates if new data has been received.
public boolean data_receive()
{
    return new_data;  // End of data_receive
}

// Method clear_data clears new data variable
public void clear_data_receive()
{
    new_data = false;
}  // End of clear_data_receive

// Method get_message retrieves data received from client for
// processing.
public void get_message(byte[] received_data)
{
    int index;
    for (index = 0; index < MAX_MESSAGE_LENGTH; index++)
    {
        received_data[index] = receiver_buffer[index];
    }  // End of get_message
}
}  // End of class Net_S_Connection
// Class Nav_Message builds and parses nav messages.
class Nav_Message implements Smart_Constants
{
  // VARIABLES for class Nav_Message
  // MSG_LENGTH - Max length for nav message is 36 bytes
  // NUM_OF_SHIP_PARMS - Nav message contains 4 parameters for ship
  // LAT_PARM_POS - Latitude parameter position in byte buffer
  // LON_PARM_POS - Longitude parameter position in byte buffer
  // CRS_PARM_POS - Course parameter position in byte buffer
  // SPD_PARM_POS - Speed parameter position in byte buffer
  // Y_POS - Ship's y position in byte buffer
  // X_POS - Ship's x position in byte buffer
  // CRS_POS - Ship's course position in byte buffer
  // SPD_POS - Ship's speed position in byte buffer
  // C_CRS_POS - Ship's command course position in byte buffer
  // C_SPD_POS - Ship's command speed position in byte buffer
  // C_ACL_POS - Ship's command acceleration in byte buffer
  // PARM_LENGTH - Parameter's length - 4 bytes
  // ZERO_BYTE - Byte of 0's
  // index2 - Used for loop variable
  // ser_number - Serial number of message
  // MAX_SERIAL - Largest serial number after which value is set to 0
  // nav_data - Byte buffer used for storing messages
  // char_digit - Character used to store a digit as a character
  final static int MSG_LENGTH = 36;
  final static int NUM_OF_SHIP_PARMS = 4;  // x pos, y pos, course and speed
  final static int LAT_PARM_POS = 4;
  final static int LON_PARM_POS = 8;
  final static int CRS_PARM_POS = 12;
  final static int SPD_PARM_POS = 16;
  final static int X_POS = 4;
  final static int Y_POS = 8;
  final static int CRS_POS = 12;
  final static int SPD_POS = 16;
  final static int C_CRS_POS = 4;
  final static int C_SPD_POS = 8;
  final static int C_ACL_POS = 12;
  final static int PARM_LENGTH = 4;
  final static byte ZERO_BYTE = 0;
  int index2;
  static byte ser_number = 0;
  static final byte MAX_SERIAL = 125;
  static byte[] nav_data = new byte[MSG_LENGTH];
  char char_digit;

  /** Nav_Message Constructor for class */
  Nav_Message()
  {
  } // Constructor with no arguments
/** Nav_Message(int,int,int,int) Constructor for class - 4 parameters
* @param srt_lat is starting latitude of Smart Ship.
* @param srt_lon is starting longitude of Smart Ship.
* @param srt_crs is starting course of Smart Ship.
* @param srt_spd is starting speed of Smart Ship.
*/

Nav_Message(int srt_lat, int srt_lon, int srt_crs, int srt_spd)
{
    int index,index2;
    int num_length,num_128,rem_128;

    for (index = 0; index < MSG_LENGTH; index++)
        nav_data[index] = ZERO_BYTE;                // Zero out message array

    // Code to fill in speed field of nav message
    nav_data[SPD_PARM_POS + 3] = (byte)(0x00000007f & srt_spd);

    // Code to fill in course field of nav message
    num_128 = srt_crs / 128;
    rem_128 = srt_crs % 128;
    nav_data[CRS_PARM_POS + 3] = (byte)rem_128;
    nav_data[CRS_PARM_POS + 2] = (byte)num_128;
    nav_data[MSG_TYPE_POS] = NAV_TYPE;
    nav_data[REC_TYPE_POS] = NAV_INIT;
    nav_data[SUB_REC_POS] = ZERO_BYTE;
    nav_data[SER_RET_POS] = ZERO_BYTE;
}
     // End of method Nav_Message

// Method update_msg updates nav parameters for class
void update_msg(int[] cmd_update)
{
    int index,index2,num_128,rem_128;
    int num_length;
    for (index = 0; index < MSG_LENGTH; index++)
        nav_data[index] = ZERO_BYTE;                //zero out message array

    // Code to fill in acceleration field of nav message
    nav_data[C_ACL_POS + 3] = (byte)(0x00000007f & cmd_update[2]);

    // Code to fill in speed field on nav message
    nav_data[C_SPD_POS + 3] = (byte)(0x00000007f & cmd_update[1]);

    // Code to fill in crs field of nav message
    num_128 = cmd_update[0] / 128;
    rem_128 = cmd_update[0] % 128;
    nav_data[C_CRS_POS + 3] = (byte)rem_128;
    nav_data[C_CRS_POS + 2] = (byte)num_128;
    nav_data[MSG_TYPE_POS] = NAV_TYPE;
    nav_data[REC_TYPE_POS] = NAV_CMD;
    nav_data[SUB_REC_POS] = ZERO_BYTE;
    nav_data[SER_RET_POS] = ZERO_BYTE;
void get_nav_msg(byte[] init_msg)
{
    int index;
    for (index = 0; index < MSG_LENGTH; index++)
    
        init_msg[index] = nav_data[index];
    
    // End of get_init_msg

void get_ship_init(byte[] rec_buf, int[] ship_init_info)
{
    int index, num_128, rem_128;
    ship_init_info[0] = 0;  // Zero out lat field
    ship_init_info[1] = 0;  // Zero out lon field
    ship_init_info[2] = 0;  // Zero out the course field
    ship_init_info[3] = 0;  // Zero out the speed field

    // Parse out the speed field of nav message
    ship_init_info[3] = rec_buf[SPD_PARM_POS + 3];

    // Parse out the course field of nav message
    num_128 = rec_buf[CRS_PARM_POS + 2];
    num_128 = num_128 * 128;
    rem_128 = rec_buf[CRS_PARM_POS + 3];
    ship_init_info[2] = num_128 + rem_128;

    // End of get_ship_init

void get_ship_cmd(byte[] rec_buf, int[] ship_cmd_info)
{
    int index, num_128, rem_128;
    ship_cmd_info[0] = 0;
    ship_cmd_info[1] = 0;
    ship_cmd_info[2] = 0;

    // Parse out the acceleration field of nav message
    ship_cmd_info[2] = rec_buf[C_ACL_POS + 3];

    // Parse out the speed field of nav message
    ship_cmd_info[1] = rec_buf[C_CRS_POS + 3];

    // Parse out the course field of nav message
    num_128 = rec_buf[C_SPD_POS + 2];
    num_128 = num_128 * 128;
    rem_128 = rec_buf[C_SPD_POS + 3];
    ship_cmd_info[0] = num_128 + rem_128;

    // End of get_ship_cmd

// Method build_status_msg builds status message to be sent to client
void build_status_msg(byte[] nav_data, int[] ship_status_info)
{
    int index;
    byte num_128,rem_128;
    for (index = 0; index < MSG_LENGTH; index++)
        nav_data[index] = ZERO_BYTE;  //zero out message array

    // Code to fill in spd position field of nav message
    nav_data[SPD_POS + 3] = (byte)(0x00000007f & ship_status_info[3]);

    // Code to fill in crs position field of nav message
    num_128 = (byte)(ship_status_info[2] / 128);
    rem_128 = (byte)(ship_status_info[2] % 128);
    nav_data[CRS_POS + 2] = num_128;
    nav_data[CRS_POS + 3] = rem_128;

    // Code to fill in y position field of nav message
    num_128 = (byte)(ship_status_info[1] / 128);
    rem_128 = (byte)(ship_status_info[1] % 128);
    nav_data[Y_POS + 2] = num_128;
    nav_data[Y_POS + 3] = rem_128;

    // Code to fill in x position field of nav message
    num_128 = (byte)(ship_status_info[0] / 128);
    rem_128 = (byte)(ship_status_info[0] % 128);
    nav_data[X_POS + 2] = num_128;
    nav_data[X_POS + 3] = rem_128;

    nav_data[MSG_TYPE_POS] = NAV_TYPE;
    nav_data[REC_TYPE_POS] = NAV_CMD;
    nav_data[SUB_REC_POS] = ZERO_BYTE;
    // Fill in serial number
    nav_data[SER_NUM_LOC] = ser_number;
    ser_number++;
    if (ser_number > MAX_SERIAL)
        ser_number = 0;
}

}  // End of build_status_msg

}  // End of class Nav_Message
public interface Smart_Constants
{
    final static int BITS_IN_BYTE = 8; // Number of bits in a byte
    final static int BYTES_IN_INT = 4; // Number of bytes in an integer
    final static int SER_RET_POS = 3; // Serial number position in buffer
    final static int MSG_TYPE_POS = 0; // Message type position in buffer
    final static int REC_TYPE_POS = 1; // Record type position in buffer
    final static int SUB_REC_POS = 2; // Sub record type position in buffer
    final static int MAX_MESSAGE_LENGTH = 72; // Maximum bytes in buffer
    final static int MAX_ARRAY = 25;
    final static int SERVER_PORT = 1998; // Serial port used for sockets
    final static int REC_SUB_TYPE_POS = 1; // Sub record type position in buffer
    final static byte NAV_TYPE = 1; // NAV_TYPE value
    final static byte NAV_INIT = 1; // NAV_INIT value
    final static byte NAV_CMD = 2; // NAV_CMD value
    final static byte NAV_STAT = 3; // NAV_STAT value
    final static byte SER_NUM_LOC = 3; // SER_NUM_LOC type position in buffer
    final static byte MODEL_STOP = 99; // MODEL_STOP value
}
  // End of Smart_Constants
**APPENDIX O.  EXPERIMENT 1 DATA FOR TCP**

Experiment 1 (Server Control using TCP, Windows NT and Internet Explorer)

<table>
<thead>
<tr>
<th>N</th>
<th>stat_u.txt Message Counter</th>
<th>stat_u.txt Message Counter</th>
<th>sent_u.txt Message Counter</th>
<th>sent_u.txt Message Counter</th>
<th>Round Trip Latency (ms)</th>
<th>Calculated one way Latency (ms)</th>
<th>Freq. Bin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 868731829592</td>
<td>0 868731829512</td>
<td>1 868731831655</td>
<td>2 868731833768</td>
<td>3 868731835891</td>
<td>4 868731836004</td>
<td>5 868731840127</td>
</tr>
<tr>
<td></td>
<td></td>
<td>868732026705</td>
<td>93</td>
<td>868732026675</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>--------------</td>
<td>----</td>
<td>--------------</td>
<td>----</td>
<td>----</td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>93</td>
<td>868732028818</td>
<td>94</td>
<td>868732028798</td>
<td>40</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>95</td>
<td>868732030911</td>
<td>95</td>
<td>868732030911</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>96</td>
<td>868732033334</td>
<td>96</td>
<td>868732033034</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>97</td>
<td>97</td>
<td>868732035187</td>
<td>97</td>
<td>868732035157</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>98</td>
<td>98</td>
<td>868732037340</td>
<td>98</td>
<td>868732037270</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>99</td>
<td>868732039494</td>
<td>99</td>
<td>868732039423</td>
<td>71</td>
<td>35.5</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>868732041546</td>
<td>100</td>
<td>868732041536</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>101</td>
<td>868732043720</td>
<td>101</td>
<td>868732043650</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>102</td>
<td>868732045873</td>
<td>102</td>
<td>868732045773</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>103</td>
<td>868732047926</td>
<td>103</td>
<td>868732047886</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>104</td>
<td>868732050099</td>
<td>104</td>
<td>868732050009</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>105</td>
<td>868732052242</td>
<td>105</td>
<td>868732052142</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>106</td>
<td>868732054305</td>
<td>106</td>
<td>868732054265</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>107</td>
<td>107</td>
<td>868732056458</td>
<td>107</td>
<td>868732056378</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>108</td>
<td>108</td>
<td>868732058521</td>
<td>108</td>
<td>868732058491</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>109</td>
<td>109</td>
<td>868732060684</td>
<td>109</td>
<td>868732060614</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>110</td>
<td>868732062817</td>
<td>110</td>
<td>868732062727</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>111</td>
<td>868732064940</td>
<td>111</td>
<td>868732064850</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>112</td>
<td>112</td>
<td>868732067053</td>
<td>112</td>
<td>868732066963</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>113</td>
<td>113</td>
<td>868732069186</td>
<td>113</td>
<td>868732069086</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>114</td>
<td>114</td>
<td>868732071299</td>
<td>114</td>
<td>868732071209</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>115</td>
<td>115</td>
<td>868732073412</td>
<td>115</td>
<td>868732073322</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>116</td>
<td>116</td>
<td>868732075545</td>
<td>116</td>
<td>868732075435</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>117</td>
<td>117</td>
<td>868732077588</td>
<td>117</td>
<td>868732077558</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>118</td>
<td>118</td>
<td>868732079741</td>
<td>118</td>
<td>868732079681</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>119</td>
<td>119</td>
<td>868732081804</td>
<td>119</td>
<td>868732081794</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>120</td>
<td>868732083967</td>
<td>120</td>
<td>868732083907</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>121</td>
<td>121</td>
<td>868732086121</td>
<td>121</td>
<td>868732086030</td>
<td>91</td>
<td>45.5</td>
<td></td>
</tr>
<tr>
<td>122</td>
<td>122</td>
<td>868732088184</td>
<td>122</td>
<td>868732088143</td>
<td>41</td>
<td>20.5</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>123</td>
<td>868732090337</td>
<td>123</td>
<td>868732090257</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>124</td>
<td>124</td>
<td>868732092390</td>
<td>124</td>
<td>868732092380</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>125</td>
<td>125</td>
<td>868732094553</td>
<td>125</td>
<td>868732094493</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>126</td>
<td></td>
<td>868732096706</td>
<td>0</td>
<td>868732096616</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>127</td>
<td></td>
<td>868732098759</td>
<td>1</td>
<td>868732098729</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td></td>
<td>868732100912</td>
<td>2</td>
<td>868732100852</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>129</td>
<td></td>
<td>868732103075</td>
<td>3</td>
<td>868732102985</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>130</td>
<td></td>
<td>868732105128</td>
<td>4</td>
<td>868732105098</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>131</td>
<td></td>
<td>868732107281</td>
<td>5</td>
<td>868732107221</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>132</td>
<td></td>
<td>868732109444</td>
<td>6</td>
<td>868732109344</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>133</td>
<td></td>
<td>868732111477</td>
<td>7</td>
<td>868732111457</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>134</td>
<td></td>
<td>868732113600</td>
<td>8</td>
<td>868732113570</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>135</td>
<td></td>
<td>868732115733</td>
<td>9</td>
<td>868732115693</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>136</td>
<td></td>
<td>868732117896</td>
<td>10</td>
<td>868732117816</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>137</td>
<td></td>
<td>868732120059</td>
<td>11</td>
<td>868732119969</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>138</td>
<td></td>
<td>868732122112</td>
<td>12</td>
<td>868732122082</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>139</td>
<td></td>
<td>868732124245</td>
<td>13</td>
<td>868732124195</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>140</td>
<td></td>
<td>868732126368</td>
<td>14</td>
<td>868732126318</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>141</td>
<td></td>
<td>868732128512</td>
<td>15</td>
<td>868732128431</td>
<td>81</td>
<td>40.5</td>
<td></td>
</tr>
<tr>
<td>142</td>
<td></td>
<td>868732130635</td>
<td>16</td>
<td>868732130544</td>
<td>91</td>
<td>45.5</td>
<td></td>
</tr>
<tr>
<td>143</td>
<td></td>
<td>868732132768</td>
<td>17</td>
<td>868732132668</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

159
145  18  868732134821  18  868732134791  30  15
146  19  868732136974  19  868732136904  70  35
147  20  868732139127  20  868732139037  90  45
148  21  8687321411190  21  868732141160  30  15
149  22  868732143373  22  868732143273  100  50
150  23  868732145506  23  868732145416  90  45
151  24  868732147559  24  868732147539  20  10
152  25  868732149702  25  868732149652  50  25
153  26  868732151865  26  868732151775  90  45
154  27  868732153908  27  868732153888  20  10
155  28  868732156051  28  868732156011  40  20
156  29  868732158134  29  868732158124  10  5
157  30  868732160297  30  868732160237  60  30
158  31  868732162460  31  868732162360  100  50
159  32  868732164503  32  868732164473  30  15
160  33  868732166636  33  868732166596  40  20
161  34  868732168759  34  868732168709  50  25
162  35  868732170902  35  868732170822  80  40
163  36  868732173036  36  868732172945  91  45.5
164  37  868732175159  37  868732175068  91  45.5
165  38  868732177212  38  868732177182  30  15
166  39  868732179385  39  868732179305  60  30
167  40  868732181518  40  868732181418  100  50
168  41  868732183571  41  868732183541  30  15
169  42  868732185734  42  868732185654  80  40
170  43  868732187787  43  868732187777  10  5
171  44  868732189950  44  868732189890  60  30
172  45  868732192073  45  868732192003  70  35
173  46  868732194226  46  868732194126  100  50
174  47  868732196279  47  868732196259  20  10
175  48  868732198442  48  868732198372  70  35
176  49  868732200585  49  868732200495  90  45
177  50  868732202658  50  868732202608  50  25
178  51  868732204811  51  868732204731  80  40
179  52  868732206874  52  868732206844  30  15
180  53  868732209007  53  868732208957  50  25
181  54  868732211110  54  868732211080  30  15
182  55  868732213233  55  868732213193  40  20
183  56  868732215356  56  868732215306  50  25
184  57  868732217479  57  868732217419  60  30
185  58  868732219613  58  868732219532  81  40.5
186  59  868732221736  59  868732221645  91  45.5
187  60  868732223839  60  868732223769  70  35
188  61  868732225962  61  868732225882  80  40
189  62  868732228015  62  868732227995  20  10
190  63  868732230178  63  868732230108  70  35
191  64  868732232341  64  868732232241  100  50
192  65  868732234454  65  868732234354  100  50
193  66  868732236577  66  868732236477  100  50
194  67  868732238700  67  868732238610  90  45
195  68  868732240823  68  868732240733  90  45
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>196</td>
<td>69</td>
<td>868732242946</td>
<td>69</td>
<td>868732242846</td>
<td>100</td>
</tr>
<tr>
<td>197</td>
<td>70</td>
<td>868732245069</td>
<td>70</td>
<td>868732244969</td>
<td>100</td>
</tr>
<tr>
<td>198</td>
<td>71</td>
<td>868732247122</td>
<td>71</td>
<td>868732247082</td>
<td>40</td>
</tr>
<tr>
<td>199</td>
<td>72</td>
<td>868732249285</td>
<td>72</td>
<td>868732249215</td>
<td>70</td>
</tr>
<tr>
<td>200</td>
<td>73</td>
<td>868732251348</td>
<td>73</td>
<td>868732251338</td>
<td>10</td>
</tr>
<tr>
<td>201</td>
<td>74</td>
<td>868732253501</td>
<td>74</td>
<td>868732253451</td>
<td>50</td>
</tr>
<tr>
<td>202</td>
<td>75</td>
<td>868732255654</td>
<td>75</td>
<td>868732255564</td>
<td>90</td>
</tr>
<tr>
<td>203</td>
<td>76</td>
<td>868732257707</td>
<td>76</td>
<td>868732257677</td>
<td>30</td>
</tr>
<tr>
<td>204</td>
<td>77</td>
<td>868732259850</td>
<td>77</td>
<td>868732259790</td>
<td>60</td>
</tr>
<tr>
<td>205</td>
<td>78</td>
<td>868732262014</td>
<td>78</td>
<td>868732261913</td>
<td>101</td>
</tr>
<tr>
<td>206</td>
<td>79</td>
<td>868732264046</td>
<td>79</td>
<td>868732264026</td>
<td>20</td>
</tr>
<tr>
<td>207</td>
<td>80</td>
<td>868732266149</td>
<td>80</td>
<td>868732266139</td>
<td>10</td>
</tr>
<tr>
<td>208</td>
<td>81</td>
<td>868732268283</td>
<td>81</td>
<td>868732268262</td>
<td>21</td>
</tr>
<tr>
<td>209</td>
<td>82</td>
<td>868732270416</td>
<td>82</td>
<td>868732270376</td>
<td>40</td>
</tr>
<tr>
<td>210</td>
<td>83</td>
<td>868732272529</td>
<td>83</td>
<td>868732272489</td>
<td>40</td>
</tr>
<tr>
<td>211</td>
<td>84</td>
<td>868732274662</td>
<td>84</td>
<td>868732274602</td>
<td>60</td>
</tr>
<tr>
<td>212</td>
<td>85</td>
<td>868732276835</td>
<td>85</td>
<td>868732276735</td>
<td>100</td>
</tr>
<tr>
<td>213</td>
<td>86</td>
<td>868732278888</td>
<td>86</td>
<td>868732278848</td>
<td>40</td>
</tr>
<tr>
<td>214</td>
<td>87</td>
<td>868732281051</td>
<td>87</td>
<td>868732280971</td>
<td>80</td>
</tr>
<tr>
<td>215</td>
<td>88</td>
<td>868732283214</td>
<td>88</td>
<td>868732283114</td>
<td>100</td>
</tr>
<tr>
<td>216</td>
<td>89</td>
<td>868732285267</td>
<td>89</td>
<td>868732285227</td>
<td>40</td>
</tr>
<tr>
<td>217</td>
<td>90</td>
<td>868732287420</td>
<td>90</td>
<td>868732287340</td>
<td>80</td>
</tr>
<tr>
<td>218</td>
<td>91</td>
<td>868732289563</td>
<td>91</td>
<td>868732289463</td>
<td>100</td>
</tr>
<tr>
<td>219</td>
<td>92</td>
<td>868732291616</td>
<td>92</td>
<td>868732291576</td>
<td>40</td>
</tr>
<tr>
<td>220</td>
<td>93</td>
<td>868732293769</td>
<td>93</td>
<td>868732293689</td>
<td>80</td>
</tr>
<tr>
<td>221</td>
<td>94</td>
<td>868732295882</td>
<td>94</td>
<td>868732295812</td>
<td>70</td>
</tr>
<tr>
<td>222</td>
<td>95</td>
<td>868732298005</td>
<td>95</td>
<td>868732297925</td>
<td>80</td>
</tr>
<tr>
<td>223</td>
<td>96</td>
<td>868732300148</td>
<td>96</td>
<td>868732300058</td>
<td>90</td>
</tr>
<tr>
<td>224</td>
<td>97</td>
<td>868732302201</td>
<td>97</td>
<td>868732302171</td>
<td>30</td>
</tr>
<tr>
<td>225</td>
<td>98</td>
<td>868732304364</td>
<td>98</td>
<td>868732304284</td>
<td>80</td>
</tr>
<tr>
<td>226</td>
<td>99</td>
<td>868732306507</td>
<td>99</td>
<td>868732306417</td>
<td>90</td>
</tr>
<tr>
<td>227</td>
<td>100</td>
<td>868732308631</td>
<td>100</td>
<td>868732308530</td>
<td>101</td>
</tr>
<tr>
<td>228</td>
<td>101</td>
<td>868732310734</td>
<td>101</td>
<td>868732310643</td>
<td>91</td>
</tr>
<tr>
<td>229</td>
<td>102</td>
<td>868732312867</td>
<td>102</td>
<td>868732312756</td>
<td>111</td>
</tr>
<tr>
<td>230</td>
<td>103</td>
<td>868732314900</td>
<td>103</td>
<td>868732314890</td>
<td>10</td>
</tr>
<tr>
<td>231</td>
<td>104</td>
<td>868732317013</td>
<td>104</td>
<td>868732317003</td>
<td>10</td>
</tr>
<tr>
<td>232</td>
<td>105</td>
<td>868732319146</td>
<td>105</td>
<td>868732319116</td>
<td>30</td>
</tr>
<tr>
<td>233</td>
<td>106</td>
<td>868732321269</td>
<td>106</td>
<td>868732321229</td>
<td>40</td>
</tr>
<tr>
<td>234</td>
<td>107</td>
<td>868732323392</td>
<td>107</td>
<td>868732323352</td>
<td>40</td>
</tr>
<tr>
<td>235</td>
<td>108</td>
<td>868732325535</td>
<td>108</td>
<td>868732325465</td>
<td>70</td>
</tr>
<tr>
<td>236</td>
<td>109</td>
<td>868732327698</td>
<td>109</td>
<td>868732327598</td>
<td>100</td>
</tr>
<tr>
<td>237</td>
<td>110</td>
<td>868732329751</td>
<td>110</td>
<td>868732329711</td>
<td>40</td>
</tr>
<tr>
<td>238</td>
<td>111</td>
<td>868732331884</td>
<td>111</td>
<td>868732331834</td>
<td>50</td>
</tr>
<tr>
<td>239</td>
<td>112</td>
<td>868732333997</td>
<td>112</td>
<td>868732333947</td>
<td>50</td>
</tr>
<tr>
<td>240</td>
<td>113</td>
<td>868732336150</td>
<td>113</td>
<td>868732336060</td>
<td>90</td>
</tr>
<tr>
<td>241</td>
<td>114</td>
<td>868732338193</td>
<td>114</td>
<td>868732338183</td>
<td>10</td>
</tr>
<tr>
<td>242</td>
<td>115</td>
<td>868732340346</td>
<td>115</td>
<td>868732340296</td>
<td>50</td>
</tr>
<tr>
<td>243</td>
<td>116</td>
<td>868732342499</td>
<td>116</td>
<td>868732342409</td>
<td>90</td>
</tr>
<tr>
<td>244</td>
<td>117</td>
<td>868732344542</td>
<td>117</td>
<td>868732344542</td>
<td>100</td>
</tr>
<tr>
<td>245</td>
<td>118</td>
<td>868732346685</td>
<td>118</td>
<td>868732346655</td>
<td>30</td>
</tr>
<tr>
<td>246</td>
<td>119</td>
<td>868732348858</td>
<td>119</td>
<td>868732348768</td>
<td>90</td>
</tr>
</tbody>
</table>

161
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>100</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>247</td>
<td>120</td>
<td>868732351001</td>
<td>120</td>
<td>868732350901</td>
<td></td>
<td></td>
</tr>
<tr>
<td>248</td>
<td>121</td>
<td>868732353074</td>
<td>121</td>
<td>868732353014</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td>249</td>
<td>122</td>
<td>868732355208</td>
<td>122</td>
<td>868732355127</td>
<td>81</td>
<td>40.5</td>
</tr>
<tr>
<td>250</td>
<td>123</td>
<td>868732357381</td>
<td>123</td>
<td>868732357311</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>251</td>
<td>124</td>
<td>868732359444</td>
<td>124</td>
<td>868732359424</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>252</td>
<td>125</td>
<td>868732361587</td>
<td>125</td>
<td>868732361547</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>253</td>
<td>0</td>
<td>868732363700</td>
<td>0</td>
<td>868732363660</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>254</td>
<td>1</td>
<td>868732365853</td>
<td>1</td>
<td>868732365773</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>255</td>
<td>2</td>
<td>868732368016</td>
<td>2</td>
<td>868732367926</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>256</td>
<td>3</td>
<td>868732370079</td>
<td>3</td>
<td>868732370039</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>257</td>
<td>4</td>
<td>868732372242</td>
<td>4</td>
<td>868732372152</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>258</td>
<td>5</td>
<td>868732374375</td>
<td>5</td>
<td>868732374285</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>259</td>
<td>6</td>
<td>868732376488</td>
<td>6</td>
<td>868732376398</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>260</td>
<td>7</td>
<td>868732378531</td>
<td>7</td>
<td>868732378521</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>261</td>
<td>8</td>
<td>868732380704</td>
<td>8</td>
<td>868732380654</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>262</td>
<td>9</td>
<td>868732382857</td>
<td>9</td>
<td>868732382767</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>263</td>
<td>10</td>
<td>868732384910</td>
<td>10</td>
<td>868732384890</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>264</td>
<td>11</td>
<td>868732387043</td>
<td>11</td>
<td>868732387013</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>265</td>
<td>12</td>
<td>868732389166</td>
<td>12</td>
<td>868732389126</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>266</td>
<td>13</td>
<td>868732391299</td>
<td>13</td>
<td>868732391239</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td>267</td>
<td>14</td>
<td>868732393422</td>
<td>14</td>
<td>868732393372</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>268</td>
<td>15</td>
<td>868732395546</td>
<td>15</td>
<td>868732395485</td>
<td>61</td>
<td>30.5</td>
</tr>
<tr>
<td>269</td>
<td>16</td>
<td>868732397699</td>
<td>16</td>
<td>868732397598</td>
<td>101</td>
<td>50.5</td>
</tr>
<tr>
<td>270</td>
<td>17</td>
<td>868732399762</td>
<td>17</td>
<td>868732399712</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>271</td>
<td>18</td>
<td>868732401925</td>
<td>18</td>
<td>868732401835</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>272</td>
<td>19</td>
<td>868732404078</td>
<td>19</td>
<td>868732403988</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>273</td>
<td>20</td>
<td>868732406191</td>
<td>20</td>
<td>868732406101</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>274</td>
<td>21</td>
<td>868732408304</td>
<td>21</td>
<td>868732408304</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>275</td>
<td>22</td>
<td>868732410457</td>
<td>22</td>
<td>868732410417</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>276</td>
<td>23</td>
<td>868732412580</td>
<td>23</td>
<td>868732412540</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>277</td>
<td>24</td>
<td>868732414723</td>
<td>24</td>
<td>868732414653</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>278</td>
<td>25</td>
<td>868732416876</td>
<td>25</td>
<td>868732416786</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>279</td>
<td>26</td>
<td>868732418949</td>
<td>26</td>
<td>868732418899</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>280</td>
<td>27</td>
<td>868732421082</td>
<td>27</td>
<td>868732421012</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>281</td>
<td>28</td>
<td>868732423235</td>
<td>28</td>
<td>868732423135</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>282</td>
<td>29</td>
<td>868732425288</td>
<td>29</td>
<td>868732425248</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>283</td>
<td>30</td>
<td>868732427441</td>
<td>30</td>
<td>868732427371</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>284</td>
<td>31</td>
<td>868732429514</td>
<td>31</td>
<td>868732429484</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>285</td>
<td>32</td>
<td>868732431677</td>
<td>32</td>
<td>868732431657</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>286</td>
<td>33</td>
<td>868732433831</td>
<td>33</td>
<td>868732433730</td>
<td>101</td>
<td>50.5</td>
</tr>
<tr>
<td>287</td>
<td>34</td>
<td>868732435853</td>
<td>34</td>
<td>868732435843</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>288</td>
<td>35</td>
<td>868732437977</td>
<td>35</td>
<td>868732437956</td>
<td>21</td>
<td>10.5</td>
</tr>
<tr>
<td>289</td>
<td>36</td>
<td>868732440120</td>
<td>36</td>
<td>868732440070</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>290</td>
<td>37</td>
<td>868732442283</td>
<td>37</td>
<td>868732442183</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>291</td>
<td>38</td>
<td>868732444436</td>
<td>38</td>
<td>868732444346</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>292</td>
<td>39</td>
<td>868732446489</td>
<td>39</td>
<td>868732446459</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>293</td>
<td>40</td>
<td>868732449163</td>
<td>40</td>
<td>868732449133</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>294</td>
<td>41</td>
<td>868732451316</td>
<td>41</td>
<td>868732451266</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>295</td>
<td>42</td>
<td>868732453479</td>
<td>42</td>
<td>868732453379</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>296</td>
<td>43</td>
<td>868732455532</td>
<td>43</td>
<td>868732455502</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>297</td>
<td>44</td>
<td>868732457685</td>
<td>44</td>
<td>868732457625</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>349</td>
<td>96</td>
<td>868732567853</td>
<td>96</td>
<td>868732567753</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>350</td>
<td>97</td>
<td>868732569896</td>
<td>97</td>
<td>868732569866</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>351</td>
<td>98</td>
<td>868732572009</td>
<td>98</td>
<td>868732571979</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>352</td>
<td>99</td>
<td>868732574242</td>
<td>99</td>
<td>868732574092</td>
<td>25</td>
<td>15</td>
</tr>
<tr>
<td>353</td>
<td>100</td>
<td>868732576275</td>
<td>100</td>
<td>868732576205</td>
<td>20</td>
<td>15</td>
</tr>
<tr>
<td>354</td>
<td>101</td>
<td>868732578368</td>
<td>101</td>
<td>868732578328</td>
<td>20</td>
<td>15</td>
</tr>
<tr>
<td>355</td>
<td>102</td>
<td>868732580001</td>
<td>102</td>
<td>868732580044</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>356</td>
<td>103</td>
<td>868732582445</td>
<td>103</td>
<td>868732582554</td>
<td>45.5</td>
<td>15</td>
</tr>
<tr>
<td>357</td>
<td>104</td>
<td>868732584697</td>
<td>104</td>
<td>868732584667</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>358</td>
<td>105</td>
<td>868732586851</td>
<td>105</td>
<td>868732586791</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>359</td>
<td>106</td>
<td>868732588984</td>
<td>106</td>
<td>868732588904</td>
<td>40</td>
<td>10</td>
</tr>
<tr>
<td>360</td>
<td>107</td>
<td>868732591037</td>
<td>107</td>
<td>868732591017</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>361</td>
<td>108</td>
<td>868732593200</td>
<td>108</td>
<td>868732593140</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>362</td>
<td>109</td>
<td>868732595353</td>
<td>109</td>
<td>868732595263</td>
<td>45</td>
<td>15</td>
</tr>
<tr>
<td>363</td>
<td>110</td>
<td>868732597406</td>
<td>110</td>
<td>868732597376</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>364</td>
<td>111</td>
<td>868732599559</td>
<td>111</td>
<td>868732599489</td>
<td>35</td>
<td>10</td>
</tr>
<tr>
<td>365</td>
<td>112</td>
<td>868732601172</td>
<td>112</td>
<td>868732601222</td>
<td>45</td>
<td>15</td>
</tr>
<tr>
<td>366</td>
<td>113</td>
<td>868732603775</td>
<td>113</td>
<td>868732603735</td>
<td>20</td>
<td>15</td>
</tr>
<tr>
<td>367</td>
<td>114</td>
<td>868732605938</td>
<td>114</td>
<td>868732605984</td>
<td>45</td>
<td>15</td>
</tr>
<tr>
<td>368</td>
<td>115</td>
<td>868732607961</td>
<td>115</td>
<td>868732607971</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>369</td>
<td>116</td>
<td>868732610144</td>
<td>116</td>
<td>868732610084</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>370</td>
<td>117</td>
<td>868732612307</td>
<td>117</td>
<td>868732612207</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>371</td>
<td>118</td>
<td>868732614360</td>
<td>118</td>
<td>868732614330</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>372</td>
<td>119</td>
<td>868732616513</td>
<td>119</td>
<td>868732616443</td>
<td>35</td>
<td>15</td>
</tr>
<tr>
<td>373</td>
<td>120</td>
<td>868732618656</td>
<td>120</td>
<td>868732618556</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>374</td>
<td>121</td>
<td>868732620749</td>
<td>121</td>
<td>868732620669</td>
<td>40</td>
<td>25</td>
</tr>
<tr>
<td>375</td>
<td>122</td>
<td>868732622892</td>
<td>122</td>
<td>868732622792</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>376</td>
<td>123</td>
<td>868732624955</td>
<td>123</td>
<td>868732624905</td>
<td>25</td>
<td>15</td>
</tr>
<tr>
<td>377</td>
<td>124</td>
<td>868732627108</td>
<td>124</td>
<td>868732627028</td>
<td>40</td>
<td>25</td>
</tr>
<tr>
<td>378</td>
<td>125</td>
<td>868732629272</td>
<td>125</td>
<td>868732629191</td>
<td>40.5</td>
<td>40.5</td>
</tr>
<tr>
<td>379</td>
<td>0</td>
<td>868732631395</td>
<td>0</td>
<td>868732631305</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>380</td>
<td>1</td>
<td>868732633318</td>
<td>1</td>
<td>868732633428</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>381</td>
<td>2</td>
<td>868732635661</td>
<td>2</td>
<td>868732635661</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>382</td>
<td>3</td>
<td>868732637694</td>
<td>3</td>
<td>868732637674</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>383</td>
<td>4</td>
<td>868732639807</td>
<td>4</td>
<td>868732639787</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>384</td>
<td>5</td>
<td>868732641940</td>
<td>5</td>
<td>868732641900</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>385</td>
<td>6</td>
<td>868732644063</td>
<td>6</td>
<td>868732644023</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>386</td>
<td>7</td>
<td>868732646176</td>
<td>7</td>
<td>868732646136</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>387</td>
<td>8</td>
<td>868732648299</td>
<td>8</td>
<td>868732648249</td>
<td>25</td>
<td>10</td>
</tr>
<tr>
<td>388</td>
<td>9</td>
<td>868732650412</td>
<td>9</td>
<td>868732650362</td>
<td>25</td>
<td>10</td>
</tr>
<tr>
<td>389</td>
<td>10</td>
<td>868732652535</td>
<td>10</td>
<td>868732652485</td>
<td>25</td>
<td>10</td>
</tr>
<tr>
<td>390</td>
<td>11</td>
<td>868732654678</td>
<td>11</td>
<td>868732654608</td>
<td>35</td>
<td>15</td>
</tr>
<tr>
<td>391</td>
<td>12</td>
<td>868732656781</td>
<td>12</td>
<td>868732656721</td>
<td>45</td>
<td>15</td>
</tr>
<tr>
<td>392</td>
<td>13</td>
<td>868732658884</td>
<td>13</td>
<td>868732658834</td>
<td>20</td>
<td>15</td>
</tr>
<tr>
<td>393</td>
<td>14</td>
<td>868732661017</td>
<td>14</td>
<td>868732660947</td>
<td>35</td>
<td>15</td>
</tr>
<tr>
<td>394</td>
<td>15</td>
<td>868732663150</td>
<td>15</td>
<td>868732663060</td>
<td>45</td>
<td>15</td>
</tr>
<tr>
<td>395</td>
<td>16</td>
<td>868732665203</td>
<td>16</td>
<td>868732665183</td>
<td>20</td>
<td>15</td>
</tr>
<tr>
<td>396</td>
<td>17</td>
<td>868732667336</td>
<td>17</td>
<td>868732667296</td>
<td>20</td>
<td>15</td>
</tr>
<tr>
<td>397</td>
<td>18</td>
<td>868732669459</td>
<td>18</td>
<td>868732669409</td>
<td>25</td>
<td>15</td>
</tr>
<tr>
<td>398</td>
<td>19</td>
<td>868732671572</td>
<td>19</td>
<td>868732671522</td>
<td>25</td>
<td>15</td>
</tr>
<tr>
<td>399</td>
<td>20</td>
<td>868732673715</td>
<td>20</td>
<td>868732673645</td>
<td>35</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>868732675849</td>
<td></td>
<td>868732675758</td>
<td></td>
<td>91</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>401</td>
<td>22</td>
<td>868732677892</td>
<td>22</td>
<td>868732677781</td>
<td>21</td>
<td>10.5</td>
</tr>
<tr>
<td>402</td>
<td>23</td>
<td>868732680035</td>
<td>23</td>
<td>868732679985</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>403</td>
<td>24</td>
<td>868732682158</td>
<td>24</td>
<td>868732682108</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>404</td>
<td>25</td>
<td>868732684281</td>
<td>25</td>
<td>868732684221</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td>405</td>
<td>26</td>
<td>868732686434</td>
<td>26</td>
<td>868732686334</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>406</td>
<td>27</td>
<td>868732688467</td>
<td>27</td>
<td>868732688447</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>407</td>
<td>28</td>
<td>868732690620</td>
<td>28</td>
<td>868732690560</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td>408</td>
<td>29</td>
<td>868732692773</td>
<td>29</td>
<td>868732692683</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>409</td>
<td>30</td>
<td>868732694826</td>
<td>30</td>
<td>868732694796</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>410</td>
<td>31</td>
<td>868732696979</td>
<td>31</td>
<td>868732696909</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>411</td>
<td>32</td>
<td>868732699132</td>
<td>32</td>
<td>868732699042</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>412</td>
<td>33</td>
<td>868732701195</td>
<td>33</td>
<td>868732701155</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>413</td>
<td>34</td>
<td>868732703358</td>
<td>34</td>
<td>868732703268</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>414</td>
<td>35</td>
<td>868732705481</td>
<td>35</td>
<td>868732705391</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>415</td>
<td>36</td>
<td>868732707534</td>
<td>36</td>
<td>868732707504</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>416</td>
<td>37</td>
<td>868732709677</td>
<td>37</td>
<td>868732709617</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td>417</td>
<td>38</td>
<td>868732711790</td>
<td>38</td>
<td>868732711740</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>418</td>
<td>39</td>
<td>868732713891</td>
<td>39</td>
<td>868732713853</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td>419</td>
<td>40</td>
<td>868732716036</td>
<td>40</td>
<td>868732715966</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>420</td>
<td>41</td>
<td>868732718159</td>
<td>41</td>
<td>868732718079</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>421</td>
<td>42</td>
<td>868732720272</td>
<td>42</td>
<td>868732720202</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>422</td>
<td>43</td>
<td>868732722426</td>
<td>43</td>
<td>868732722315</td>
<td>111</td>
<td>55.5</td>
</tr>
<tr>
<td>423</td>
<td>44</td>
<td>868732724559</td>
<td>44</td>
<td>868732724458</td>
<td>101</td>
<td>50.5</td>
</tr>
<tr>
<td>424</td>
<td>45</td>
<td>868732726612</td>
<td>45</td>
<td>868732726582</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>425</td>
<td>46</td>
<td>868732728765</td>
<td>46</td>
<td>868732728695</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>426</td>
<td>47</td>
<td>868732730898</td>
<td>47</td>
<td>868732730808</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>427</td>
<td>48</td>
<td>868732732951</td>
<td>48</td>
<td>868732732921</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>428</td>
<td>49</td>
<td>868732735124</td>
<td>49</td>
<td>868732735044</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>429</td>
<td>50</td>
<td>868732737177</td>
<td>50</td>
<td>868732737157</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>430</td>
<td>51</td>
<td>868732739340</td>
<td>51</td>
<td>868732739270</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>431</td>
<td>52</td>
<td>868732741473</td>
<td>52</td>
<td>868732741383</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>432</td>
<td>53</td>
<td>868732743596</td>
<td>53</td>
<td>868732743506</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>433</td>
<td>54</td>
<td>868732745639</td>
<td>54</td>
<td>868732745619</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>434</td>
<td>55</td>
<td>868732747782</td>
<td>55</td>
<td>868732747732</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>435</td>
<td>56</td>
<td>868732749895</td>
<td>56</td>
<td>868732749845</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>436</td>
<td>57</td>
<td>868732752028</td>
<td>57</td>
<td>868732751958</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>437</td>
<td>58</td>
<td>868732754161</td>
<td>58</td>
<td>868732754081</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>438</td>
<td>59</td>
<td>868732756284</td>
<td>59</td>
<td>868732756194</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>439</td>
<td>60</td>
<td>868732758337</td>
<td>60</td>
<td>868732758307</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>440</td>
<td>61</td>
<td>868732760500</td>
<td>61</td>
<td>868732760430</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>441</td>
<td>62</td>
<td>868732762553</td>
<td>62</td>
<td>868732762543</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>442</td>
<td>63</td>
<td>868732764716</td>
<td>63</td>
<td>868732764656</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td>443</td>
<td>64</td>
<td>868732766789</td>
<td>64</td>
<td>868732766779</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>444</td>
<td>65</td>
<td>868732768952</td>
<td>65</td>
<td>868732768889</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td>445</td>
<td>66</td>
<td>868732771106</td>
<td>66</td>
<td>868732771015</td>
<td>91</td>
<td>45.5</td>
</tr>
<tr>
<td>446</td>
<td>67</td>
<td>868732773209</td>
<td>67</td>
<td>868732773128</td>
<td>81</td>
<td>40.5</td>
</tr>
<tr>
<td>447</td>
<td>68</td>
<td>868732775352</td>
<td>68</td>
<td>868732775262</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>448</td>
<td>69</td>
<td>868732777415</td>
<td>69</td>
<td>868732777375</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>449</td>
<td>70</td>
<td>868732779528</td>
<td>70</td>
<td>868732779488</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>450</td>
<td>71</td>
<td>868732781651</td>
<td>71</td>
<td>868732781601</td>
<td>50</td>
<td>25</td>
</tr>
</tbody>
</table>

165
<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>553</td>
<td>48</td>
<td>868732999885</td>
<td>48</td>
<td>868732999854</td>
<td>31</td>
<td>15.5</td>
</tr>
<tr>
<td>554</td>
<td>49</td>
<td>868733002038</td>
<td>49</td>
<td>868733001968</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>555</td>
<td>50</td>
<td>868733004201</td>
<td>50</td>
<td>868733004101</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>556</td>
<td>51</td>
<td>868733006254</td>
<td>51</td>
<td>868733006214</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>557</td>
<td>52</td>
<td>868733008538</td>
<td>52</td>
<td>868733008337</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>558</td>
<td>53</td>
<td>868733101501</td>
<td>53</td>
<td>868733101046</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>559</td>
<td>54</td>
<td>868733126150</td>
<td>54</td>
<td>86873312573</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>560</td>
<td>55</td>
<td>868733147666</td>
<td>55</td>
<td>86873314696</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>561</td>
<td>56</td>
<td>868733168899</td>
<td>56</td>
<td>86873316809</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>562</td>
<td>57</td>
<td>868733190128</td>
<td>57</td>
<td>86873318922</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>563</td>
<td>58</td>
<td>868733211358</td>
<td>58</td>
<td>86873321045</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>564</td>
<td>59</td>
<td>868733232248</td>
<td>59</td>
<td>86873323158</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>565</td>
<td>60</td>
<td>868733253716</td>
<td>60</td>
<td>86873325281</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>566</td>
<td>61</td>
<td>868733275004</td>
<td>61</td>
<td>86873327404</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>567</td>
<td>62</td>
<td>868733295676</td>
<td>62</td>
<td>86873329517</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>568</td>
<td>63</td>
<td>868733317200</td>
<td>63</td>
<td>86873331630</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>569</td>
<td>64</td>
<td>868733338636</td>
<td>64</td>
<td>86873333773</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>570</td>
<td>65</td>
<td>868733359765</td>
<td>65</td>
<td>86873335886</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>571</td>
<td>66</td>
<td>868733380192</td>
<td>66</td>
<td>86873337999</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>572</td>
<td>67</td>
<td>868733401722</td>
<td>67</td>
<td>86873340122</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>573</td>
<td>68</td>
<td>868733423267</td>
<td>68</td>
<td>86873342235</td>
<td>91</td>
<td>45.5</td>
</tr>
<tr>
<td>574</td>
<td>69</td>
<td>868733443899</td>
<td>69</td>
<td>86873344348</td>
<td>41</td>
<td>20.5</td>
</tr>
<tr>
<td>575</td>
<td>70</td>
<td>868733465122</td>
<td>70</td>
<td>86873346461</td>
<td>51</td>
<td>25.5</td>
</tr>
<tr>
<td>576</td>
<td>71</td>
<td>868733486357</td>
<td>71</td>
<td>86873348585</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>577</td>
<td>72</td>
<td>868733507668</td>
<td>72</td>
<td>86873350698</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>578</td>
<td>73</td>
<td>868733528919</td>
<td>73</td>
<td>86873352811</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>579</td>
<td>74</td>
<td>868733550149</td>
<td>74</td>
<td>86873354924</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>580</td>
<td>75</td>
<td>868733571372</td>
<td>75</td>
<td>86873357047</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>581</td>
<td>76</td>
<td>868733591807</td>
<td>76</td>
<td>86873359160</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>582</td>
<td>77</td>
<td>868733613437</td>
<td>77</td>
<td>86873361273</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>583</td>
<td>78</td>
<td>868733634968</td>
<td>78</td>
<td>86873363396</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>584</td>
<td>79</td>
<td>868733655193</td>
<td>79</td>
<td>86873365519</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>585</td>
<td>80</td>
<td>868733677322</td>
<td>80</td>
<td>86873367642</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>586</td>
<td>81</td>
<td>868733698855</td>
<td>81</td>
<td>86873369765</td>
<td>120</td>
<td>60</td>
</tr>
<tr>
<td>587</td>
<td>82</td>
<td>868733722197</td>
<td>82</td>
<td>86873372148</td>
<td>71</td>
<td>35.5</td>
</tr>
<tr>
<td>588</td>
<td>83</td>
<td>868733743822</td>
<td>83</td>
<td>86873374281</td>
<td>101</td>
<td>50.5</td>
</tr>
<tr>
<td>589</td>
<td>84</td>
<td>868733764357</td>
<td>84</td>
<td>86873376405</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>590</td>
<td>85</td>
<td>868733785485</td>
<td>85</td>
<td>86873378518</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>591</td>
<td>86</td>
<td>868733806712</td>
<td>86</td>
<td>86873380631</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>592</td>
<td>87</td>
<td>868733827947</td>
<td>87</td>
<td>86873382744</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>593</td>
<td>88</td>
<td>868733849177</td>
<td>88</td>
<td>86873384867</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>594</td>
<td>89</td>
<td>868733870303</td>
<td>89</td>
<td>86873386980</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>595</td>
<td>90</td>
<td>868733891633</td>
<td>90</td>
<td>86873389093</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>596</td>
<td>91</td>
<td>868733912926</td>
<td>91</td>
<td>86873391226</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>597</td>
<td>92</td>
<td>868733933797</td>
<td>92</td>
<td>86873393349</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>598</td>
<td>93</td>
<td>868733955425</td>
<td>93</td>
<td>86873395462</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>599</td>
<td>94</td>
<td>868733976655</td>
<td>94</td>
<td>86873397575</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>600</td>
<td>95</td>
<td>868733998729</td>
<td>95</td>
<td>86873399688</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>601</td>
<td>96</td>
<td>868734018717</td>
<td>96</td>
<td>86873401811</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td>602</td>
<td>97</td>
<td>868734039947</td>
<td>97</td>
<td>86873403924</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>603</td>
<td>98</td>
<td>868734060978</td>
<td>98</td>
<td>86873406037</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>604</td>
<td>99</td>
<td>868733108240</td>
<td>99</td>
<td>868733108160</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>605</td>
<td>100</td>
<td>868733110293</td>
<td>100</td>
<td>868733110273</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>606</td>
<td>101</td>
<td>868733112456</td>
<td>101</td>
<td>868733112396</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td>607</td>
<td>102</td>
<td>868733114579</td>
<td>102</td>
<td>868733114509</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>608</td>
<td>103</td>
<td>868733116702</td>
<td>103</td>
<td>868733116622</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>609</td>
<td>104</td>
<td>868733118765</td>
<td>104</td>
<td>868733118735</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>610</td>
<td>105</td>
<td>868733120929</td>
<td>105</td>
<td>868733120858</td>
<td>71</td>
<td>35.5</td>
</tr>
<tr>
<td>611</td>
<td>106</td>
<td>868733123082</td>
<td>106</td>
<td>868733122982</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>612</td>
<td>107</td>
<td>868733125145</td>
<td>107</td>
<td>868733125105</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>613</td>
<td>108</td>
<td>868733127258</td>
<td>108</td>
<td>868733127218</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>614</td>
<td>109</td>
<td>868733129401</td>
<td>109</td>
<td>868733129331</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>615</td>
<td>110</td>
<td>868733131474</td>
<td>110</td>
<td>868733131444</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>616</td>
<td>111</td>
<td>868733133827</td>
<td>111</td>
<td>868733133587</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td>617</td>
<td>112</td>
<td>868733135690</td>
<td>112</td>
<td>868733135680</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>618</td>
<td>113</td>
<td>868733137843</td>
<td>113</td>
<td>868733137793</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>619</td>
<td>114</td>
<td>868733140006</td>
<td>114</td>
<td>868733139916</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>620</td>
<td>115</td>
<td>868733142059</td>
<td>115</td>
<td>868733142029</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>621</td>
<td>116</td>
<td>868733144202</td>
<td>116</td>
<td>868733144142</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td>622</td>
<td>117</td>
<td>868733146325</td>
<td>117</td>
<td>868733146255</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>623</td>
<td>118</td>
<td>868733148448</td>
<td>118</td>
<td>868733148368</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>624</td>
<td>119</td>
<td>868733150621</td>
<td>119</td>
<td>868733150521</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>625</td>
<td>120</td>
<td>868733152674</td>
<td>120</td>
<td>868733152634</td>
<td>40</td>
<td>20</td>
</tr>
</tbody>
</table>
### APPENDIX P.  EXPERIMENT 1 DATA FOR UDP

Experiment 1 (Server Control using UDP, Windows NT and Internet Explorer)

<table>
<thead>
<tr>
<th>N</th>
<th>stat_u.txt</th>
<th>stat_u.txt</th>
<th>sent_u.txt</th>
<th>sent_u..txt</th>
<th>Round Trip Latency Time</th>
<th>Calculated one way latency time</th>
<th>Freq. Bin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>868734767777</td>
<td>0</td>
<td>868734767626</td>
<td>151</td>
<td>75.5</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>868734769830</td>
<td>1</td>
<td>868734769810</td>
<td>20</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>868734771993</td>
<td>2</td>
<td>868734771923</td>
<td>70</td>
<td>35</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>868734774146</td>
<td>3</td>
<td>868734774046</td>
<td>100</td>
<td>50</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>868734776199</td>
<td>4</td>
<td>868734776159</td>
<td>40</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>868734778362</td>
<td>5</td>
<td>868734778282</td>
<td>80</td>
<td>40</td>
<td>26</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>868734780435</td>
<td>6</td>
<td>868734780395</td>
<td>40</td>
<td>20</td>
<td>31</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>868734782598</td>
<td>7</td>
<td>868734782508</td>
<td>90</td>
<td>45</td>
<td>36</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>868734784661</td>
<td>8</td>
<td>868734784621</td>
<td>40</td>
<td>20</td>
<td>41</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>868734786814</td>
<td>9</td>
<td>868734786744</td>
<td>70</td>
<td>35</td>
<td>46</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>868734788977</td>
<td>10</td>
<td>868734788877</td>
<td>100</td>
<td>50</td>
<td>51</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>868734791020</td>
<td>11</td>
<td>868734790990</td>
<td>30</td>
<td>15</td>
<td>56</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
<td>868734793183</td>
<td>12</td>
<td>868734793113</td>
<td>70</td>
<td>35</td>
<td>61</td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>868734795336</td>
<td>13</td>
<td>868734795246</td>
<td>90</td>
<td>45</td>
<td>66</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>868734797389</td>
<td>14</td>
<td>868734797359</td>
<td>30</td>
<td>15</td>
<td>71</td>
</tr>
<tr>
<td>16</td>
<td>15</td>
<td>868734799542</td>
<td>15</td>
<td>868734799472</td>
<td>70</td>
<td>35</td>
<td>76</td>
</tr>
<tr>
<td>17</td>
<td>16</td>
<td>868734801695</td>
<td>16</td>
<td>868734801605</td>
<td>90</td>
<td>45</td>
<td>81</td>
</tr>
<tr>
<td>18</td>
<td>17</td>
<td>868734803768</td>
<td>17</td>
<td>868734803718</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>18</td>
<td>868734805931</td>
<td>18</td>
<td>868734805831</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>19</td>
<td>868734807984</td>
<td>19</td>
<td>868734807944</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>20</td>
<td>868734810128</td>
<td>20</td>
<td>868734810067</td>
<td>61</td>
<td>30.5</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>21</td>
<td>868734812291</td>
<td>21</td>
<td>868734812200</td>
<td>91</td>
<td>45.5</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>22</td>
<td>868734814354</td>
<td>22</td>
<td>868734814324</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>23</td>
<td>868734816477</td>
<td>23</td>
<td>868734816437</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>24</td>
<td>868734818600</td>
<td>24</td>
<td>868734818550</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>25</td>
<td>868734820723</td>
<td>25</td>
<td>868734820663</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>26</td>
<td>868734822786</td>
<td>26</td>
<td>868734822786</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>27</td>
<td>868734825019</td>
<td>27</td>
<td>868734824929</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>28</td>
<td>868734827072</td>
<td>28</td>
<td>868734827042</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>29</td>
<td>868734829205</td>
<td>29</td>
<td>868734829155</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>868734831328</td>
<td>30</td>
<td>868734831268</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>31</td>
<td>868734833421</td>
<td>31</td>
<td>868734833391</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>32</td>
<td>868734835564</td>
<td>32</td>
<td>868734835504</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>33</td>
<td>868734837707</td>
<td>33</td>
<td>868734837617</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>34</td>
<td>868734839760</td>
<td>34</td>
<td>868734839740</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>35</td>
<td>868734841923</td>
<td>35</td>
<td>868734841853</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>36</td>
<td>868734843986</td>
<td>36</td>
<td>868734843966</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>37</td>
<td>868734846149</td>
<td>37</td>
<td>868734846089</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>38</td>
<td>868734848272</td>
<td>38</td>
<td>868734848202</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>39</td>
<td>868734850335</td>
<td>39</td>
<td>868734850315</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>40</td>
<td>868734852508</td>
<td>40</td>
<td>868734852428</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>41</td>
<td>868734854611</td>
<td>41</td>
<td>868734854551</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>42</td>
<td>868734856765</td>
<td>42</td>
<td>868734856674</td>
<td>91</td>
<td>45.5</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>43</td>
<td>868734858918</td>
<td>43</td>
<td>868734858838</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>44</td>
<td>868734861041</td>
<td>44</td>
<td>868734860951</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>868734863154</td>
<td>868734863074</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>-------------</td>
<td>-------------</td>
<td>----</td>
<td>----</td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>45</td>
<td>868734865287</td>
<td>868734865207</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>46</td>
<td>868734867410</td>
<td>868734867320</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>47</td>
<td>868734869523</td>
<td>868734869443</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>48</td>
<td>868734871646</td>
<td>868734871566</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>49</td>
<td>868734873769</td>
<td>868734873689</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>50</td>
<td>868734875892</td>
<td>868734875802</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>51</td>
<td>868734878005</td>
<td>868734877925</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>52</td>
<td>868734880128</td>
<td>868734880038</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>53</td>
<td>868734882181</td>
<td>868734882151</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>54</td>
<td>868734884214</td>
<td>868734884274</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>55</td>
<td>868734886437</td>
<td>868734886387</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>56</td>
<td>868734888580</td>
<td>868734888500</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>57</td>
<td>868734890713</td>
<td>868734890623</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>58</td>
<td>868734892826</td>
<td>868734892746</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>59</td>
<td>868734894970</td>
<td>868734894879</td>
<td>91</td>
<td>45.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>60</td>
<td>868734897022</td>
<td>868734896992</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>61</td>
<td>868734899186</td>
<td>868734899105</td>
<td>81</td>
<td>40.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>62</td>
<td>868734901329</td>
<td>868734901239</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>63</td>
<td>868734903402</td>
<td>868734903352</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>64</td>
<td>868734905555</td>
<td>868734905475</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>65</td>
<td>868734907618</td>
<td>868734907588</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>66</td>
<td>868734909771</td>
<td>868734909701</td>
<td>70</td>
<td>35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>67</td>
<td>868734911934</td>
<td>868734911834</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>68</td>
<td>868734913987</td>
<td>868734913957</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>69</td>
<td>868734916140</td>
<td>868734916070</td>
<td>70</td>
<td>35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>70</td>
<td>868734918293</td>
<td>868734918203</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>71</td>
<td>868734920436</td>
<td>868734920316</td>
<td>120</td>
<td>60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>72</td>
<td>868734922449</td>
<td>868734922429</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>73</td>
<td>868734924572</td>
<td>868734924542</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>74</td>
<td>868734926725</td>
<td>868734926665</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>75</td>
<td>868734928888</td>
<td>868734928798</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>76</td>
<td>868734930941</td>
<td>868734930911</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>77</td>
<td>868734933064</td>
<td>868734933024</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>78</td>
<td>868734935187</td>
<td>868734935147</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>79</td>
<td>868734937320</td>
<td>868734937260</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>80</td>
<td>868734939453</td>
<td>868734939373</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>81</td>
<td>868734941567</td>
<td>868734941486</td>
<td>81</td>
<td>40.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>83</td>
<td>82</td>
<td>868734943690</td>
<td>868734943609</td>
<td>81</td>
<td>40.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>83</td>
<td>868734945813</td>
<td>868734945722</td>
<td>91</td>
<td>45.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>84</td>
<td>868734947936</td>
<td>868734947846</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>86</td>
<td>85</td>
<td>868734950069</td>
<td>868734949979</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>87</td>
<td>86</td>
<td>868734952122</td>
<td>868734952102</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>87</td>
<td>868734954285</td>
<td>868734954215</td>
<td>70</td>
<td>35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>89</td>
<td>88</td>
<td>868734956438</td>
<td>868734956348</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>89</td>
<td>868734958491</td>
<td>868734958461</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>90</td>
<td>868734960654</td>
<td>868734960584</td>
<td>70</td>
<td>35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>91</td>
<td>868734962807</td>
<td>868734962727</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>93</td>
<td>92</td>
<td>868734964880</td>
<td>868734964840</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>93</td>
<td>868734967013</td>
<td>868734966953</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>94</td>
<td>868734969146</td>
<td>868734969066</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>97</td>
<td>96</td>
<td>868734971269</td>
<td>96</td>
<td>868734971189</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>98</td>
<td>97</td>
<td>868734973332</td>
<td>97</td>
<td>868734973302</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>98</td>
<td>868734975495</td>
<td>98</td>
<td>868734975415</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>99</td>
<td>868734977758</td>
<td>99</td>
<td>868734977738</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>100</td>
<td>868734979721</td>
<td>100</td>
<td>868734979651</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>101</td>
<td>868734981874</td>
<td>101</td>
<td>868734981784</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>102</td>
<td>868734983937</td>
<td>102</td>
<td>868734983897</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>103</td>
<td>868734986101</td>
<td>103</td>
<td>868734986020</td>
<td>81</td>
<td>40.5</td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>104</td>
<td>868734988164</td>
<td>104</td>
<td>868734988133</td>
<td>31</td>
<td>15.5</td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>105</td>
<td>868734990327</td>
<td>105</td>
<td>868734990247</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>107</td>
<td>106</td>
<td>868734992490</td>
<td>106</td>
<td>868734992400</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>108</td>
<td>107</td>
<td>868734994613</td>
<td>107</td>
<td>868734994513</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>109</td>
<td>108</td>
<td>868734996666</td>
<td>108</td>
<td>868734996636</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>109</td>
<td>868734998809</td>
<td>109</td>
<td>868734998749</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>110</td>
<td>868735000932</td>
<td>110</td>
<td>868735000862</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>112</td>
<td>111</td>
<td>868735003045</td>
<td>111</td>
<td>868735002985</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>113</td>
<td>112</td>
<td>868735005168</td>
<td>112</td>
<td>868735005098</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>114</td>
<td>113</td>
<td>868735007281</td>
<td>113</td>
<td>868735007211</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>115</td>
<td>114</td>
<td>868735009414</td>
<td>114</td>
<td>868735009334</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>116</td>
<td>115</td>
<td>868735011467</td>
<td>115</td>
<td>868735011447</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>117</td>
<td>116</td>
<td>868735013580</td>
<td>116</td>
<td>868735013560</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>118</td>
<td>117</td>
<td>868735015733</td>
<td>117</td>
<td>868735015683</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>119</td>
<td>118</td>
<td>868735017866</td>
<td>118</td>
<td>868735017796</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>119</td>
<td>868735019979</td>
<td>119</td>
<td>868735019909</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>121</td>
<td>120</td>
<td>868735022132</td>
<td>120</td>
<td>868735022042</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>122</td>
<td>121</td>
<td>868735024165</td>
<td>121</td>
<td>868735024155</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>122</td>
<td>868735026288</td>
<td>122</td>
<td>868735026268</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>124</td>
<td>123</td>
<td>868735028441</td>
<td>123</td>
<td>868735028391</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>125</td>
<td>124</td>
<td>868735030595</td>
<td>124</td>
<td>868735030504</td>
<td>91</td>
<td>45.5</td>
<td></td>
</tr>
<tr>
<td>126</td>
<td>125</td>
<td>868735032647</td>
<td>125</td>
<td>868735032617</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>127</td>
<td>126</td>
<td>868735034801</td>
<td>126</td>
<td>868735034730</td>
<td>71</td>
<td>35.5</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>127</td>
<td>868735036864</td>
<td>127</td>
<td>868735036854</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>129</td>
<td>128</td>
<td>868735039017</td>
<td>128</td>
<td>868735038967</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>129</td>
<td>868735041180</td>
<td>129</td>
<td>868735041090</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>131</td>
<td>130</td>
<td>868735043223</td>
<td>130</td>
<td>868735043203</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>132</td>
<td>131</td>
<td>868735045386</td>
<td>131</td>
<td>868735045316</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>133</td>
<td>132</td>
<td>868735047539</td>
<td>132</td>
<td>868735047449</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>134</td>
<td>133</td>
<td>868735049592</td>
<td>133</td>
<td>868735049562</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>135</td>
<td>134</td>
<td>868735051745</td>
<td>134</td>
<td>868735051685</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>136</td>
<td>135</td>
<td>868735053898</td>
<td>135</td>
<td>868735053808</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>137</td>
<td>136</td>
<td>868735055951</td>
<td>136</td>
<td>868735055921</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>138</td>
<td>137</td>
<td>868735058114</td>
<td>137</td>
<td>868735058034</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>139</td>
<td>138</td>
<td>868735060167</td>
<td>138</td>
<td>868735060157</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>139</td>
<td>868735062330</td>
<td>139</td>
<td>868735062270</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>141</td>
<td>140</td>
<td>868735064483</td>
<td>140</td>
<td>868735064393</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>142</td>
<td>141</td>
<td>868735066606</td>
<td>141</td>
<td>868735066526</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>143</td>
<td>142</td>
<td>868735068729</td>
<td>142</td>
<td>868735068639</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>144</td>
<td>143</td>
<td>868735070872</td>
<td>143</td>
<td>868735070772</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>145</td>
<td>144</td>
<td>868735072975</td>
<td>144</td>
<td>868735072895</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>146</td>
<td>145</td>
<td>868735075099</td>
<td>145</td>
<td>868735075018</td>
<td>81</td>
<td>40.5</td>
<td></td>
</tr>
<tr>
<td>147</td>
<td>146</td>
<td>868735077232</td>
<td>146</td>
<td>868735077141</td>
<td>91</td>
<td>45.5</td>
<td></td>
</tr>
<tr>
<td>148</td>
<td>21</td>
<td>868735079275</td>
<td>21</td>
<td>868735079254</td>
<td>21</td>
<td>10.5</td>
<td></td>
</tr>
<tr>
<td>149</td>
<td>22</td>
<td>868735081438</td>
<td>22</td>
<td>868735081378</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>23</td>
<td>868735083591</td>
<td>23</td>
<td>868735083521</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>151</td>
<td>24</td>
<td>868735085704</td>
<td>24</td>
<td>868735085634</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>152</td>
<td>25</td>
<td>868735087827</td>
<td>25</td>
<td>868735087747</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>153</td>
<td>26</td>
<td>868735089970</td>
<td>26</td>
<td>868735089870</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>154</td>
<td>27</td>
<td>868735092083</td>
<td>27</td>
<td>868735091993</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>155</td>
<td>28</td>
<td>868735094206</td>
<td>28</td>
<td>868735094116</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>156</td>
<td>29</td>
<td>868735096329</td>
<td>29</td>
<td>868735096239</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>157</td>
<td>30</td>
<td>868735098392</td>
<td>30</td>
<td>868735098352</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>158</td>
<td>31</td>
<td>868735100545</td>
<td>31</td>
<td>868735100465</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>159</td>
<td>32</td>
<td>868735102618</td>
<td>32</td>
<td>868735102578</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>33</td>
<td>868735104771</td>
<td>33</td>
<td>868735104701</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>161</td>
<td>34</td>
<td>868735106834</td>
<td>34</td>
<td>868735106834</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>162</td>
<td>35</td>
<td>868735108957</td>
<td>35</td>
<td>868735108947</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>163</td>
<td>36</td>
<td>868735111080</td>
<td>36</td>
<td>868735111060</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>164</td>
<td>37</td>
<td>868735113243</td>
<td>37</td>
<td>868735113173</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>165</td>
<td>38</td>
<td>868735115386</td>
<td>38</td>
<td>868735115306</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>166</td>
<td>39</td>
<td>868735117540</td>
<td>39</td>
<td>868735117449</td>
<td>91</td>
<td>45.5</td>
<td></td>
</tr>
<tr>
<td>167</td>
<td>40</td>
<td>868735119603</td>
<td>40</td>
<td>868735119562</td>
<td>41</td>
<td>20.5</td>
<td></td>
</tr>
<tr>
<td>168</td>
<td>41</td>
<td>868735121756</td>
<td>41</td>
<td>868735121675</td>
<td>81</td>
<td>40.5</td>
<td></td>
</tr>
<tr>
<td>169</td>
<td>42</td>
<td>868735123819</td>
<td>42</td>
<td>868735123799</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>170</td>
<td>43</td>
<td>868735125992</td>
<td>43</td>
<td>868735125912</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>171</td>
<td>44</td>
<td>868735128145</td>
<td>44</td>
<td>868735128055</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>172</td>
<td>45</td>
<td>868735130198</td>
<td>45</td>
<td>868735130168</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>173</td>
<td>46</td>
<td>868735132361</td>
<td>46</td>
<td>868735132281</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>47</td>
<td>868735134474</td>
<td>47</td>
<td>868735134404</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>175</td>
<td>48</td>
<td>868735136617</td>
<td>48</td>
<td>868735136527</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>176</td>
<td>49</td>
<td>868735138760</td>
<td>49</td>
<td>868735138670</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>177</td>
<td>50</td>
<td>868735140813</td>
<td>50</td>
<td>868735140783</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>178</td>
<td>51</td>
<td>868735142976</td>
<td>51</td>
<td>868735142906</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>179</td>
<td>52</td>
<td>868735145129</td>
<td>52</td>
<td>868735145029</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>180</td>
<td>53</td>
<td>868735147242</td>
<td>53</td>
<td>868735147152</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>181</td>
<td>54</td>
<td>868735149365</td>
<td>54</td>
<td>868735149265</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>182</td>
<td>55</td>
<td>868735151508</td>
<td>55</td>
<td>868735151408</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>183</td>
<td>56</td>
<td>868735153621</td>
<td>56</td>
<td>868735153531</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>184</td>
<td>57</td>
<td>868735155734</td>
<td>57</td>
<td>868735155644</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>185</td>
<td>58</td>
<td>868735157787</td>
<td>58</td>
<td>868735157767</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>186</td>
<td>59</td>
<td>868735159995</td>
<td>59</td>
<td>868735159880</td>
<td>71</td>
<td>35.5</td>
<td></td>
</tr>
<tr>
<td>187</td>
<td>60</td>
<td>868735162094</td>
<td>60</td>
<td>868735161993</td>
<td>101</td>
<td>50.5</td>
<td></td>
</tr>
<tr>
<td>188</td>
<td>61</td>
<td>868735164227</td>
<td>61</td>
<td>868735164117</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>189</td>
<td>62</td>
<td>868735166250</td>
<td>62</td>
<td>868735166230</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>190</td>
<td>63</td>
<td>868735168403</td>
<td>63</td>
<td>868735168343</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>191</td>
<td>64</td>
<td>868735170526</td>
<td>64</td>
<td>868735170466</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>192</td>
<td>65</td>
<td>868735172649</td>
<td>65</td>
<td>868735172579</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>193</td>
<td>66</td>
<td>868735174802</td>
<td>66</td>
<td>868735174712</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>194</td>
<td>67</td>
<td>868735176925</td>
<td>67</td>
<td>868735176835</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>195</td>
<td>68</td>
<td>868735179048</td>
<td>68</td>
<td>868735179848</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>196</td>
<td>69</td>
<td>868735181111</td>
<td>69</td>
<td>868735181071</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>197</td>
<td>70</td>
<td>868735183254</td>
<td>70</td>
<td>868735183184</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>198</td>
<td>71</td>
<td>868735185417</td>
<td>71</td>
<td>868735185317</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>199</td>
<td>72</td>
<td>868735187480</td>
<td>72</td>
<td>868735187440</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>73</td>
<td>868735189633</td>
<td>73</td>
<td>868735189553</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>201</td>
<td>74</td>
<td>868735191786</td>
<td>74</td>
<td>868735191696</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>202</td>
<td>75</td>
<td>868735193859</td>
<td>75</td>
<td>868735193809</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>203</td>
<td>76</td>
<td>868735196012</td>
<td>76</td>
<td>868735195922</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>204</td>
<td>77</td>
<td>868735198175</td>
<td>77</td>
<td>868735198075</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>205</td>
<td>78</td>
<td>868735200208</td>
<td>78</td>
<td>868735200188</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>206</td>
<td>79</td>
<td>868735202331</td>
<td>79</td>
<td>868735202301</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>207</td>
<td>80</td>
<td>868735204485</td>
<td>80</td>
<td>868735204424</td>
<td>61</td>
<td>30.5</td>
<td></td>
</tr>
<tr>
<td>208</td>
<td>81</td>
<td>868735206608</td>
<td>81</td>
<td>868735206538</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>209</td>
<td>82</td>
<td>868735208731</td>
<td>82</td>
<td>868735208651</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>210</td>
<td>83</td>
<td>868735210784</td>
<td>83</td>
<td>868735210774</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>211</td>
<td>84</td>
<td>868735212917</td>
<td>84</td>
<td>868735212887</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>212</td>
<td>85</td>
<td>868735215040</td>
<td>85</td>
<td>868735215000</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>213</td>
<td>86</td>
<td>868735217183</td>
<td>86</td>
<td>868735217113</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>214</td>
<td>87</td>
<td>868735219316</td>
<td>87</td>
<td>868735219236</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>215</td>
<td>88</td>
<td>868735221439</td>
<td>88</td>
<td>868735221349</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>216</td>
<td>89</td>
<td>868735223482</td>
<td>89</td>
<td>868735223362</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>217</td>
<td>90</td>
<td>868735225545</td>
<td>90</td>
<td>868735225545</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>218</td>
<td>91</td>
<td>868735227788</td>
<td>91</td>
<td>868735227698</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>219</td>
<td>92</td>
<td>868735229861</td>
<td>92</td>
<td>868735229811</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>220</td>
<td>93</td>
<td>868735231924</td>
<td>93</td>
<td>868735231924</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>221</td>
<td>94</td>
<td>868735234077</td>
<td>94</td>
<td>868735234047</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>222</td>
<td>95</td>
<td>868735236240</td>
<td>95</td>
<td>868735236160</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>223</td>
<td>96</td>
<td>868735238293</td>
<td>96</td>
<td>868735238273</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>224</td>
<td>97</td>
<td>868735240456</td>
<td>97</td>
<td>868735240386</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>225</td>
<td>98</td>
<td>868735242599</td>
<td>98</td>
<td>868735242509</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>226</td>
<td>99</td>
<td>868735244712</td>
<td>99</td>
<td>868735244622</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>227</td>
<td>100</td>
<td>868735246855</td>
<td>100</td>
<td>868735246745</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>228</td>
<td>101</td>
<td>868735248918</td>
<td>101</td>
<td>868735248868</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>229</td>
<td>102</td>
<td>868735250172</td>
<td>102</td>
<td>868735250981</td>
<td>91</td>
<td>45.5</td>
<td></td>
</tr>
<tr>
<td>230</td>
<td>103</td>
<td>868735252312</td>
<td>103</td>
<td>868735253094</td>
<td>31</td>
<td>15.5</td>
<td></td>
</tr>
<tr>
<td>231</td>
<td>104</td>
<td>868735254528</td>
<td>104</td>
<td>868735255218</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>232</td>
<td>105</td>
<td>868735256742</td>
<td>105</td>
<td>868735257331</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>233</td>
<td>106</td>
<td>868735258946</td>
<td>106</td>
<td>868735259444</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>234</td>
<td>107</td>
<td>868735261161</td>
<td>107</td>
<td>868735261557</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>235</td>
<td>108</td>
<td>868735263370</td>
<td>108</td>
<td>868735263680</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>236</td>
<td>109</td>
<td>868735265583</td>
<td>109</td>
<td>868735265793</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>237</td>
<td>110</td>
<td>868735267796</td>
<td>110</td>
<td>868735267916</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>238</td>
<td>111</td>
<td>868735270099</td>
<td>111</td>
<td>868735270029</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>239</td>
<td>112</td>
<td>868735272162</td>
<td>112</td>
<td>868735272142</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>240</td>
<td>113</td>
<td>868735274325</td>
<td>113</td>
<td>868735274265</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>241</td>
<td>114</td>
<td>868735276478</td>
<td>114</td>
<td>868735276388</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>242</td>
<td>115</td>
<td>868735278631</td>
<td>115</td>
<td>868735278511</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>243</td>
<td>116</td>
<td>868735280654</td>
<td>116</td>
<td>868735280624</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>244</td>
<td>117</td>
<td>868735282797</td>
<td>117</td>
<td>868735282737</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>245</td>
<td>118</td>
<td>868735284940</td>
<td>118</td>
<td>868735284850</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>246</td>
<td>119</td>
<td>868735287063</td>
<td>119</td>
<td>868735286963</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>247</td>
<td>120</td>
<td>868735289196</td>
<td>120</td>
<td>868735289086</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>248</td>
<td>121</td>
<td>868735291329</td>
<td>121</td>
<td>868735291199</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>249</td>
<td>122</td>
<td>868735293402</td>
<td>122</td>
<td>868735293312</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
</tbody>
</table>

---

175
### APPENDIX Q. EXPERIMENT 2 DATA FOR TCP

Experiment 2 (Client Control using TCP, Windows NT and Internet Explorer)

<table>
<thead>
<tr>
<th>N</th>
<th>stat_u.txt</th>
<th>stat_u.txt</th>
<th>sent_u.txt</th>
<th>sent_u.txt</th>
<th>Round Trip Latency Time</th>
<th>Calculated one way Latency time</th>
<th>Freq. Bin</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Message Counter</td>
<td>Message Counter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0 868738526882</td>
<td>0 868738526802</td>
<td>80</td>
<td>40</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1 868738528945</td>
<td>1 868738528925</td>
<td>20</td>
<td>10</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2 868738531068</td>
<td>2 868738531038</td>
<td>30</td>
<td>15</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3 868738533181</td>
<td>3 868738533161</td>
<td>20</td>
<td>10</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4 868738535304</td>
<td>4 868738535274</td>
<td>30</td>
<td>15</td>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5 868738537417</td>
<td>5 868738537387</td>
<td>30</td>
<td>15</td>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6 868738539540</td>
<td>6 868738539510</td>
<td>30</td>
<td>15</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>7 868738541653</td>
<td>7 868738541623</td>
<td>30</td>
<td>15</td>
<td>36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>8 868738543766</td>
<td>8 868738543736</td>
<td>30</td>
<td>15</td>
<td>41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>9 868738545879</td>
<td>9 868738545859</td>
<td>20</td>
<td>10</td>
<td>46</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>10 868738547992</td>
<td>10 868738547972</td>
<td>20</td>
<td>10</td>
<td>51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>11 868738550115</td>
<td>11 868738550095</td>
<td>20</td>
<td>10</td>
<td>56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>12 868738552228</td>
<td>12 868738552208</td>
<td>20</td>
<td>10</td>
<td>61</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>13 868738554351</td>
<td>13 868738554321</td>
<td>30</td>
<td>15</td>
<td>66</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14 868738556474</td>
<td>14 868738556444</td>
<td>30</td>
<td>15</td>
<td>71</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>15 868738558588</td>
<td>15 868738558557</td>
<td>31</td>
<td>15.5</td>
<td>76</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>16 868738560701</td>
<td>16 868738560671</td>
<td>30</td>
<td>15</td>
<td>81</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>17 868738562814</td>
<td>17 868738562794</td>
<td>20</td>
<td>10</td>
<td>86</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>18 868738564937</td>
<td>18 868738564907</td>
<td>30</td>
<td>15</td>
<td>91</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>19 868738567050</td>
<td>19 868738567020</td>
<td>30</td>
<td>15</td>
<td>95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>20 868738569173</td>
<td>20 868738569143</td>
<td>30</td>
<td>15</td>
<td>99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>21 868738571276</td>
<td>21 868738571256</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>22 868738573389</td>
<td>22 868738573379</td>
<td>10</td>
<td>5</td>
<td>37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>23 868738575592</td>
<td>23 868738575492</td>
<td>100</td>
<td>50</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>24 868738577695</td>
<td>24 868738577605</td>
<td>90</td>
<td>45</td>
<td>39</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>25 868738579818</td>
<td>25 868738579728</td>
<td>90</td>
<td>45</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>26 868738581931</td>
<td>26 868738581841</td>
<td>90</td>
<td>45</td>
<td>41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>27 868738583954</td>
<td>27 868738583924</td>
<td>90</td>
<td>45</td>
<td>42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>28 868738586157</td>
<td>28 868738586077</td>
<td>80</td>
<td>40</td>
<td>43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>29 868738588220</td>
<td>29 868738588190</td>
<td>30</td>
<td>15</td>
<td>44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30 868738590343</td>
<td>30 868738590303</td>
<td>40</td>
<td>20</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>31 868738592506</td>
<td>31 868738592426</td>
<td>80</td>
<td>40</td>
<td>46</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>32 868738594639</td>
<td>32 868738594593</td>
<td>100</td>
<td>50</td>
<td>47</td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>33 868738596752</td>
<td>33 868738596662</td>
<td>90</td>
<td>45</td>
<td>48</td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>34 868738598815</td>
<td>34 868738598775</td>
<td>40</td>
<td>20</td>
<td>49</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>35 868738600928</td>
<td>35 868738600898</td>
<td>30</td>
<td>15</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>36 868738603061</td>
<td>36 868738603011</td>
<td>50</td>
<td>25</td>
<td>51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>37 868738605215</td>
<td>37 868738605124</td>
<td>91</td>
<td>45.5</td>
<td>52</td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>38 868738607318</td>
<td>38 868738607237</td>
<td>81</td>
<td>40.5</td>
<td>53</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>39 868738609441</td>
<td>39 868738609361</td>
<td>80</td>
<td>40</td>
<td>54</td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>40 868738611564</td>
<td>40 868738611474</td>
<td>90</td>
<td>45</td>
<td>55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>41 868738613677</td>
<td>41 868738613597</td>
<td>80</td>
<td>40</td>
<td>56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>42 868738615790</td>
<td>42 868738615710</td>
<td>80</td>
<td>40</td>
<td>57</td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>43</td>
<td>868738617893</td>
<td>43</td>
<td>868738617823</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>44</td>
<td>868738620006</td>
<td>44</td>
<td>868738619946</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>45</td>
<td>868738622159</td>
<td>45</td>
<td>868738622059</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>46</td>
<td>868738624192</td>
<td>46</td>
<td>868738624182</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>47</td>
<td>868738626395</td>
<td>47</td>
<td>868738626295</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>48</td>
<td>868738628498</td>
<td>48</td>
<td>868738628408</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>49</td>
<td>868738630611</td>
<td>49</td>
<td>868738630531</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>50</td>
<td>868738632724</td>
<td>50</td>
<td>868738632644</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>51</td>
<td>868738634857</td>
<td>51</td>
<td>868738634757</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>52</td>
<td>868738636970</td>
<td>52</td>
<td>868738636880</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>53</td>
<td>868738639093</td>
<td>53</td>
<td>868738638993</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>54</td>
<td>868738641196</td>
<td>54</td>
<td>868738641106</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>55</td>
<td>868738643309</td>
<td>55</td>
<td>868738643229</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>56</td>
<td>868738645432</td>
<td>56</td>
<td>868738645342</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>57</td>
<td>868738647565</td>
<td>57</td>
<td>868738647465</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>58</td>
<td>868738649679</td>
<td>58</td>
<td>868738649588</td>
<td>91</td>
<td>45.5</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>59</td>
<td>868738651702</td>
<td>59</td>
<td>868738651701</td>
<td>101</td>
<td>50.5</td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>60</td>
<td>868738653875</td>
<td>60</td>
<td>868738653814</td>
<td>61</td>
<td>30.5</td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>61</td>
<td>868738655998</td>
<td>61</td>
<td>868738655938</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>62</td>
<td>868738658111</td>
<td>62</td>
<td>868738658051</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>63</td>
<td>868738660224</td>
<td>63</td>
<td>868738660174</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>64</td>
<td>868738662327</td>
<td>64</td>
<td>868738662287</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>65</td>
<td>868738664460</td>
<td>65</td>
<td>868738664400</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>66</td>
<td>868738666573</td>
<td>66</td>
<td>868738666523</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>67</td>
<td>868738668686</td>
<td>67</td>
<td>868738668636</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>68</td>
<td>868738670859</td>
<td>68</td>
<td>868738670749</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>69</td>
<td>868738672882</td>
<td>69</td>
<td>868738672872</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>70</td>
<td>868738675005</td>
<td>70</td>
<td>868738674985</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>71</td>
<td>868738677218</td>
<td>71</td>
<td>868738677098</td>
<td>120</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>72</td>
<td>868738679311</td>
<td>72</td>
<td>868738679221</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>73</td>
<td>868738681434</td>
<td>73</td>
<td>868738681334</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>74</td>
<td>868738683517</td>
<td>74</td>
<td>868738683457</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>75</td>
<td>868738685620</td>
<td>75</td>
<td>868738685570</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>76</td>
<td>868738687733</td>
<td>76</td>
<td>868738687683</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>77</td>
<td>868738689886</td>
<td>77</td>
<td>868738689806</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>78</td>
<td>868738691989</td>
<td>78</td>
<td>868738691919</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>79</td>
<td>868738694102</td>
<td>79</td>
<td>868738694032</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>80</td>
<td>868738696205</td>
<td>80</td>
<td>868738696155</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>81</td>
<td>868738698318</td>
<td>81</td>
<td>868738698278</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>83</td>
<td>82</td>
<td>868738700431</td>
<td>82</td>
<td>868738700391</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>83</td>
<td>868738702575</td>
<td>83</td>
<td>868738702504</td>
<td>71</td>
<td>35.5</td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>84</td>
<td>868738704688</td>
<td>84</td>
<td>868738704618</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>86</td>
<td>85</td>
<td>868738706801</td>
<td>85</td>
<td>868738706741</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>87</td>
<td>86</td>
<td>868738708894</td>
<td>86</td>
<td>868738708854</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>87</td>
<td>868738711007</td>
<td>87</td>
<td>868738710977</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>89</td>
<td>88</td>
<td>868738713130</td>
<td>88</td>
<td>868738713090</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>89</td>
<td>868738715233</td>
<td>89</td>
<td>868738715203</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>90</td>
<td>868738717346</td>
<td>90</td>
<td>868738717326</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>91</td>
<td>868738719469</td>
<td>91</td>
<td>868738719439</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>93</td>
<td>92</td>
<td>868738721582</td>
<td>92</td>
<td>868738721562</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>93</td>
<td>868738723715</td>
<td>93</td>
<td>868738723675</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>94</td>
<td>868738725828</td>
<td>94</td>
<td>868738725788</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>95</td>
<td>868738727941</td>
<td>95</td>
<td>868738727911</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>97</td>
<td>96</td>
<td>868738730104</td>
<td>96</td>
<td>868738730024</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>98</td>
<td>97</td>
<td>868738732227</td>
<td>97</td>
<td>868738732137</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>98</td>
<td>868738734350</td>
<td>98</td>
<td>868738734260</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>99</td>
<td>868738736473</td>
<td>99</td>
<td>868738736373</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>100</td>
<td>868738738576</td>
<td>100</td>
<td>868738738486</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>101</td>
<td>868738740699</td>
<td>101</td>
<td>868738740609</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>102</td>
<td>868738742802</td>
<td>102</td>
<td>868738742722</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>103</td>
<td>868738744925</td>
<td>103</td>
<td>868738744845</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>104</td>
<td>868738747039</td>
<td>104</td>
<td>868738746958</td>
<td>81</td>
<td>40.5</td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>105</td>
<td>868738749152</td>
<td>105</td>
<td>868738749071</td>
<td>81</td>
<td>40.5</td>
<td></td>
</tr>
<tr>
<td>107</td>
<td>106</td>
<td>868738751255</td>
<td>106</td>
<td>868738751184</td>
<td>71</td>
<td>35.5</td>
<td></td>
</tr>
<tr>
<td>108</td>
<td>107</td>
<td>868738753378</td>
<td>107</td>
<td>868738753308</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>109</td>
<td>108</td>
<td>868738755451</td>
<td>108</td>
<td>868738755421</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>109</td>
<td>868738757574</td>
<td>109</td>
<td>868738757534</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>110</td>
<td>868738759687</td>
<td>110</td>
<td>868738759657</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>112</td>
<td>111</td>
<td>868738761790</td>
<td>111</td>
<td>868738761770</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>113</td>
<td>112</td>
<td>868738763933</td>
<td>112</td>
<td>868738763893</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>114</td>
<td>113</td>
<td>868738766096</td>
<td>113</td>
<td>868738766006</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>115</td>
<td>114</td>
<td>868738768199</td>
<td>114</td>
<td>868738768119</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>116</td>
<td>115</td>
<td>868738770312</td>
<td>115</td>
<td>868738770242</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>117</td>
<td>116</td>
<td>868738772425</td>
<td>116</td>
<td>868738772355</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>118</td>
<td>117</td>
<td>868738774528</td>
<td>117</td>
<td>868738774468</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>119</td>
<td>118</td>
<td>868738776631</td>
<td>118</td>
<td>868738776591</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>119</td>
<td>868738778754</td>
<td>119</td>
<td>868738778704</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>121</td>
<td>120</td>
<td>868738780857</td>
<td>120</td>
<td>868738780817</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>122</td>
<td>121</td>
<td>868738782990</td>
<td>121</td>
<td>868738782940</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>122</td>
<td>868738785153</td>
<td>122</td>
<td>868738785053</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>124</td>
<td>123</td>
<td>868738787266</td>
<td>123</td>
<td>868738787176</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>125</td>
<td>124</td>
<td>868738789399</td>
<td>124</td>
<td>868738789289</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>126</td>
<td>125</td>
<td>868738791502</td>
<td>125</td>
<td>868738791412</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>127</td>
<td>0</td>
<td>868738793625</td>
<td>0</td>
<td>868738793525</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>1</td>
<td>868738795739</td>
<td>1</td>
<td>868738795648</td>
<td>91</td>
<td>45.5</td>
<td></td>
</tr>
<tr>
<td>129</td>
<td>2</td>
<td>868738797842</td>
<td>2</td>
<td>868738797761</td>
<td>81</td>
<td>40.5</td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>3</td>
<td>868738799955</td>
<td>3</td>
<td>868738799874</td>
<td>81</td>
<td>40.5</td>
<td></td>
</tr>
<tr>
<td>131</td>
<td>4</td>
<td>868738802088</td>
<td>4</td>
<td>868738801998</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>132</td>
<td>5</td>
<td>868738804191</td>
<td>5</td>
<td>868738804111</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>133</td>
<td>6</td>
<td>868738806304</td>
<td>6</td>
<td>868738806224</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>134</td>
<td>7</td>
<td>868738808457</td>
<td>7</td>
<td>868738808347</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>135</td>
<td>8</td>
<td>868738810490</td>
<td>8</td>
<td>868738810460</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>136</td>
<td>9</td>
<td>868738812603</td>
<td>9</td>
<td>868738812583</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>137</td>
<td>10</td>
<td>868738814706</td>
<td>10</td>
<td>868738814696</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>138</td>
<td>11</td>
<td>868738816829</td>
<td>11</td>
<td>868738816809</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>139</td>
<td>12</td>
<td>868738819032</td>
<td>12</td>
<td>868738818922</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>13</td>
<td>868738821135</td>
<td>13</td>
<td>868738821045</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>141</td>
<td>14</td>
<td>868738823238</td>
<td>14</td>
<td>868738823158</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>142</td>
<td>15</td>
<td>868738825341</td>
<td>15</td>
<td>868738825271</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>143</td>
<td>16</td>
<td>868738827464</td>
<td>16</td>
<td>868738827394</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>144</td>
<td>17</td>
<td>868738829567</td>
<td>17</td>
<td>868738829507</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>145</td>
<td>18</td>
<td>868738831680</td>
<td>18</td>
<td>868738831630</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>146</td>
<td>19</td>
<td>868738833793</td>
<td>19</td>
<td>868738833743</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>147</td>
<td>20</td>
<td>868738835906</td>
<td>20</td>
<td>868738835856</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>148</td>
<td>21</td>
<td>868738838039</td>
<td>21</td>
<td>868738837979</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>149</td>
<td>22</td>
<td>868738840122</td>
<td>22</td>
<td>868738840092</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>23</td>
<td>868738842235</td>
<td>23</td>
<td>868738842205</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>151</td>
<td>24</td>
<td>868738844399</td>
<td>24</td>
<td>868738844328</td>
<td>71</td>
<td>35.5</td>
<td></td>
</tr>
<tr>
<td>152</td>
<td>25</td>
<td>868738846512</td>
<td>25</td>
<td>868738846441</td>
<td>71</td>
<td>35.5</td>
<td></td>
</tr>
<tr>
<td>153</td>
<td>26</td>
<td>868738848625</td>
<td>26</td>
<td>868738848554</td>
<td>71</td>
<td>35.5</td>
<td></td>
</tr>
<tr>
<td>154</td>
<td>27</td>
<td>868738850748</td>
<td>27</td>
<td>868738850678</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>155</td>
<td>28</td>
<td>868738852861</td>
<td>28</td>
<td>868738852791</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>156</td>
<td>29</td>
<td>868738854994</td>
<td>29</td>
<td>868738854904</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>157</td>
<td>30</td>
<td>868738857107</td>
<td>30</td>
<td>868738857027</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>158</td>
<td>31</td>
<td>868738859220</td>
<td>31</td>
<td>868738859140</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>159</td>
<td>32</td>
<td>868738861333</td>
<td>32</td>
<td>868738861253</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>33</td>
<td>868738863406</td>
<td>33</td>
<td>868738863376</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>161</td>
<td>34</td>
<td>868738865529</td>
<td>34</td>
<td>868738865489</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>162</td>
<td>35</td>
<td>868738867642</td>
<td>35</td>
<td>868738867602</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>163</td>
<td>36</td>
<td>868738869735</td>
<td>36</td>
<td>8687388696725</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>164</td>
<td>37</td>
<td>868738871948</td>
<td>37</td>
<td>868738871838</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>165</td>
<td>38</td>
<td>868738874051</td>
<td>38</td>
<td>868738873951</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>166</td>
<td>39</td>
<td>868738876164</td>
<td>39</td>
<td>868738876074</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>167</td>
<td>40</td>
<td>868738878277</td>
<td>40</td>
<td>868738878187</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>168</td>
<td>41</td>
<td>868738880330</td>
<td>41</td>
<td>868738880310</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>169</td>
<td>42</td>
<td>868738882443</td>
<td>42</td>
<td>868738882423</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>170</td>
<td>43</td>
<td>868738884556</td>
<td>43</td>
<td>868738884536</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>171</td>
<td>44</td>
<td>868738886759</td>
<td>44</td>
<td>868738886659</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>172</td>
<td>45</td>
<td>868738888862</td>
<td>45</td>
<td>868738888772</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>173</td>
<td>46</td>
<td>868738890975</td>
<td>46</td>
<td>868738890885</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>47</td>
<td>868738893109</td>
<td>47</td>
<td>868738893008</td>
<td>101</td>
<td>50.5</td>
<td></td>
</tr>
<tr>
<td>175</td>
<td>48</td>
<td>868738895222</td>
<td>48</td>
<td>868738895121</td>
<td>101</td>
<td>50.5</td>
<td></td>
</tr>
<tr>
<td>176</td>
<td>49</td>
<td>868738897345</td>
<td>49</td>
<td>868738897244</td>
<td>101</td>
<td>50.5</td>
<td></td>
</tr>
<tr>
<td>177</td>
<td>50</td>
<td>868738899408</td>
<td>50</td>
<td>868738899358</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>178</td>
<td>51</td>
<td>868738901541</td>
<td>51</td>
<td>868738901471</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>179</td>
<td>52</td>
<td>868738903664</td>
<td>52</td>
<td>868738903594</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>180</td>
<td>53</td>
<td>868738905767</td>
<td>53</td>
<td>868738905707</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>181</td>
<td>54</td>
<td>868738907890</td>
<td>54</td>
<td>868738907820</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>182</td>
<td>55</td>
<td>868738910003</td>
<td>55</td>
<td>868738909943</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>183</td>
<td>56</td>
<td>868738912126</td>
<td>56</td>
<td>868738912056</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>184</td>
<td>57</td>
<td>868738914229</td>
<td>57</td>
<td>868738914179</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>185</td>
<td>58</td>
<td>868738916342</td>
<td>58</td>
<td>868738916292</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>186</td>
<td>59</td>
<td>868738918445</td>
<td>59</td>
<td>868738918405</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>187</td>
<td>60</td>
<td>868738920558</td>
<td>60</td>
<td>868738920518</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>188</td>
<td>61</td>
<td>868738922673</td>
<td>61</td>
<td>868738922641</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>189</td>
<td>62</td>
<td>868738924854</td>
<td>62</td>
<td>868738924754</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>190</td>
<td>63</td>
<td>868738926977</td>
<td>63</td>
<td>868738926877</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>191</td>
<td>64</td>
<td>868738928990</td>
<td>64</td>
<td>868738928990</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>192</td>
<td>65</td>
<td>868738931203</td>
<td>65</td>
<td>868738931113</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>193</td>
<td>66</td>
<td>868738933326</td>
<td>66</td>
<td>868738933226</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>194</td>
<td>67</td>
<td>868738935429</td>
<td>67</td>
<td>868738935349</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>195</td>
<td>68</td>
<td>868738937532</td>
<td>68</td>
<td>868738937462</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>196</td>
<td>69</td>
<td>868738939655</td>
<td>69</td>
<td>868738939575</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td></td>
<td>48</td>
<td></td>
<td>868739049724</td>
<td>121</td>
<td>121</td>
<td>868739049654</td>
<td>70</td>
</tr>
<tr>
<td>---</td>
<td>----</td>
<td>---</td>
<td>-------------</td>
<td>-----</td>
<td>-----</td>
<td>-------------</td>
<td>-----</td>
</tr>
<tr>
<td>249</td>
<td>122</td>
<td>868739051847</td>
<td>122</td>
<td>868739051767</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>123</td>
<td>868739053960</td>
<td>123</td>
<td>868739053880</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>251</td>
<td>124</td>
<td>868739056073</td>
<td>124</td>
<td>868739056003</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>252</td>
<td>125</td>
<td>868739058196</td>
<td>125</td>
<td>868739058116</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>253</td>
<td>0</td>
<td>868739060309</td>
<td>0</td>
<td>868739060229</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>254</td>
<td>1</td>
<td>868739062422</td>
<td>1</td>
<td>868739062352</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>255</td>
<td>2</td>
<td>868739064545</td>
<td>2</td>
<td>868739064465</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>3</td>
<td>868739066688</td>
<td>3</td>
<td>8687390666578</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>257</td>
<td>4</td>
<td>868739068801</td>
<td>4</td>
<td>868739068701</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>258</td>
<td>5</td>
<td>868739070904</td>
<td>5</td>
<td>868739070814</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>259</td>
<td>6</td>
<td>868739072917</td>
<td>6</td>
<td>868739072937</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>260</td>
<td>7</td>
<td>868739075150</td>
<td>7</td>
<td>868739075050</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>261</td>
<td>8</td>
<td>868739077203</td>
<td>8</td>
<td>868739077173</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>262</td>
<td>9</td>
<td>868739079336</td>
<td>9</td>
<td>868739079286</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>263</td>
<td>10</td>
<td>868739081459</td>
<td>10</td>
<td>868739081409</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>264</td>
<td>11</td>
<td>868739083572</td>
<td>11</td>
<td>868739083522</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>265</td>
<td>12</td>
<td>868739085665</td>
<td>12</td>
<td>868739085635</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>13</td>
<td>868739087778</td>
<td>13</td>
<td>868739087758</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>267</td>
<td>14</td>
<td>868739089881</td>
<td>14</td>
<td>868739089871</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>268</td>
<td>15</td>
<td>868739092005</td>
<td>15</td>
<td>868739091985</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>269</td>
<td>16</td>
<td>868739094148</td>
<td>16</td>
<td>868739094108</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>270</td>
<td>17</td>
<td>868739096261</td>
<td>17</td>
<td>868739096221</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>271</td>
<td>18</td>
<td>868739098364</td>
<td>18</td>
<td>868739098334</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>272</td>
<td>19</td>
<td>868739100487</td>
<td>19</td>
<td>868739100457</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>273</td>
<td>20</td>
<td>868739102590</td>
<td>20</td>
<td>868739102570</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>274</td>
<td>21</td>
<td>868739104713</td>
<td>21</td>
<td>868739104683</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>275</td>
<td>22</td>
<td>868739106916</td>
<td>22</td>
<td>868739106806</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>276</td>
<td>23</td>
<td>868739109019</td>
<td>23</td>
<td>868739108919</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>277</td>
<td>24</td>
<td>868739111132</td>
<td>24</td>
<td>868739111032</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>278</td>
<td>25</td>
<td>868739113255</td>
<td>25</td>
<td>868739113155</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>279</td>
<td>26</td>
<td>868739115358</td>
<td>26</td>
<td>868739115268</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>280</td>
<td>27</td>
<td>868739117481</td>
<td>27</td>
<td>868739117391</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>281</td>
<td>28</td>
<td>868739119554</td>
<td>28</td>
<td>868739119504</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>282</td>
<td>29</td>
<td>868739121677</td>
<td>29</td>
<td>868739121617</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>283</td>
<td>30</td>
<td>868739123800</td>
<td>30</td>
<td>868739123740</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>284</td>
<td>31</td>
<td>868739125873</td>
<td>31</td>
<td>868739125853</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>285</td>
<td>32</td>
<td>868739127996</td>
<td>32</td>
<td>868739127966</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>286</td>
<td>33</td>
<td>868739130019</td>
<td>33</td>
<td>868739130089</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>287</td>
<td>34</td>
<td>868739132232</td>
<td>34</td>
<td>868739132202</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>288</td>
<td>35</td>
<td>868739134345</td>
<td>35</td>
<td>868739134315</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>289</td>
<td>36</td>
<td>868739136489</td>
<td>36</td>
<td>868739136438</td>
<td>51</td>
<td>25.5</td>
<td></td>
</tr>
<tr>
<td>290</td>
<td>37</td>
<td>868739138602</td>
<td>37</td>
<td>868739138551</td>
<td>51</td>
<td>25.5</td>
<td></td>
</tr>
<tr>
<td>291</td>
<td>38</td>
<td>868739140715</td>
<td>38</td>
<td>868739140685</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>292</td>
<td>39</td>
<td>868739142878</td>
<td>39</td>
<td>868739142878</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>293</td>
<td>40</td>
<td>868739145001</td>
<td>40</td>
<td>868739144931</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>294</td>
<td>41</td>
<td>868739147134</td>
<td>41</td>
<td>868739147044</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>295</td>
<td>42</td>
<td>868739149237</td>
<td>42</td>
<td>868739149157</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>296</td>
<td>43</td>
<td>868739151350</td>
<td>43</td>
<td>868739151280</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>297</td>
<td>44</td>
<td>868739153463</td>
<td>44</td>
<td>868739153403</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>298</td>
<td>45</td>
<td>868739155576</td>
<td>45</td>
<td>868739155516</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>299</td>
<td>46</td>
<td>686739157679</td>
<td>46</td>
<td>686739157629</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>-----</td>
<td>---------------</td>
<td>-----</td>
<td>---------------</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>47</td>
<td>686739159792</td>
<td>47</td>
<td>686739159752</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>301</td>
<td>48</td>
<td>686739161905</td>
<td>48</td>
<td>686739161865</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>302</td>
<td>49</td>
<td>686739164008</td>
<td>49</td>
<td>686739163978</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>303</td>
<td>50</td>
<td>686739166141</td>
<td>50</td>
<td>686739166101</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>304</td>
<td>51</td>
<td>686739168284</td>
<td>51</td>
<td>686739168214</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>305</td>
<td>52</td>
<td>686739170407</td>
<td>52</td>
<td>686739170327</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>306</td>
<td>53</td>
<td>686739172510</td>
<td>53</td>
<td>686739172450</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>307</td>
<td>54</td>
<td>686739174613</td>
<td>54</td>
<td>686739174563</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>308</td>
<td>55</td>
<td>686739176746</td>
<td>55</td>
<td>686739176676</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>309</td>
<td>56</td>
<td>686739178889</td>
<td>56</td>
<td>686739178799</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>310</td>
<td>57</td>
<td>686739181003</td>
<td>57</td>
<td>686739180912</td>
<td>91</td>
<td>45.5</td>
</tr>
<tr>
<td></td>
<td>311</td>
<td>58</td>
<td>686739183126</td>
<td>58</td>
<td>686739183025</td>
<td>101</td>
<td>50.5</td>
</tr>
<tr>
<td></td>
<td>312</td>
<td>59</td>
<td>686739185249</td>
<td>59</td>
<td>686739185148</td>
<td>101</td>
<td>50.5</td>
</tr>
<tr>
<td></td>
<td>313</td>
<td>60</td>
<td>686739187352</td>
<td>60</td>
<td>686739187262</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>314</td>
<td>61</td>
<td>686739189475</td>
<td>61</td>
<td>686739189385</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>315</td>
<td>62</td>
<td>686739191528</td>
<td>62</td>
<td>686739191498</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>316</td>
<td>63</td>
<td>686739193641</td>
<td>63</td>
<td>686739193611</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>317</td>
<td>64</td>
<td>686739195754</td>
<td>64</td>
<td>686739195734</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>318</td>
<td>65</td>
<td>686739197867</td>
<td>65</td>
<td>686739197847</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>319</td>
<td>66</td>
<td>686739199980</td>
<td>66</td>
<td>686739199960</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>320</td>
<td>67</td>
<td>686739202093</td>
<td>67</td>
<td>686739202083</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>321</td>
<td>68</td>
<td>686739204306</td>
<td>68</td>
<td>686739204196</td>
<td>110</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>322</td>
<td>69</td>
<td>686739206409</td>
<td>69</td>
<td>686739206319</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>323</td>
<td>70</td>
<td>686739208512</td>
<td>70</td>
<td>686739208432</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>324</td>
<td>71</td>
<td>686739210615</td>
<td>71</td>
<td>686739210555</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>325</td>
<td>72</td>
<td>686739212728</td>
<td>72</td>
<td>686739212668</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>326</td>
<td>73</td>
<td>686739214811</td>
<td>73</td>
<td>686739214781</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>327</td>
<td>74</td>
<td>686739216944</td>
<td>74</td>
<td>686739216904</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>328</td>
<td>75</td>
<td>686739219067</td>
<td>75</td>
<td>686739219017</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>329</td>
<td>76</td>
<td>686739221180</td>
<td>76</td>
<td>686739221130</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>330</td>
<td>77</td>
<td>686739223293</td>
<td>77</td>
<td>686739223253</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>331</td>
<td>78</td>
<td>686739225406</td>
<td>78</td>
<td>686739225376</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>332</td>
<td>79</td>
<td>686739227509</td>
<td>79</td>
<td>686739227489</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>333</td>
<td>80</td>
<td>686739229622</td>
<td>80</td>
<td>686739229602</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>334</td>
<td>81</td>
<td>686739231745</td>
<td>81</td>
<td>686739231725</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>335</td>
<td>82</td>
<td>686739233869</td>
<td>82</td>
<td>686739233838</td>
<td>31</td>
<td>15.5</td>
</tr>
<tr>
<td></td>
<td>336</td>
<td>83</td>
<td>686739235972</td>
<td>83</td>
<td>686739235952</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>337</td>
<td>84</td>
<td>686739238175</td>
<td>84</td>
<td>686739238075</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>338</td>
<td>85</td>
<td>686739240288</td>
<td>85</td>
<td>686739240188</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>339</td>
<td>86</td>
<td>686739242411</td>
<td>86</td>
<td>686739242311</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>340</td>
<td>87</td>
<td>686739244564</td>
<td>87</td>
<td>686739244424</td>
<td>140</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>341</td>
<td>88</td>
<td>686739246587</td>
<td>88</td>
<td>686739246537</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>342</td>
<td>89</td>
<td>686739248710</td>
<td>89</td>
<td>686739248660</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>343</td>
<td>90</td>
<td>686739250813</td>
<td>90</td>
<td>686739250773</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>344</td>
<td>91</td>
<td>686739252926</td>
<td>91</td>
<td>686739252886</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>345</td>
<td>92</td>
<td>686739255029</td>
<td>92</td>
<td>686739255009</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>346</td>
<td>93</td>
<td>686739257142</td>
<td>93</td>
<td>686739257122</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>347</td>
<td>94</td>
<td>686739259345</td>
<td>94</td>
<td>686739259245</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>348</td>
<td>95</td>
<td>686739261378</td>
<td>95</td>
<td>686739261358</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>349</td>
<td>96</td>
<td>686739263501</td>
<td>96</td>
<td>686739263481</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>350</td>
<td>97</td>
<td>868739265614</td>
<td>97</td>
<td>868739265594</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>351</td>
<td>98</td>
<td>868739267737</td>
<td>98</td>
<td>868739267707</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>352</td>
<td>99</td>
<td>868739269850</td>
<td>99</td>
<td>868739269830</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>353</td>
<td>100</td>
<td>868739271963</td>
<td>100</td>
<td>868739271943</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>354</td>
<td>101</td>
<td>868739274086</td>
<td>101</td>
<td>868739274056</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>355</td>
<td>102</td>
<td>868739276199</td>
<td>102</td>
<td>868739276179</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>356</td>
<td>103</td>
<td>868739278312</td>
<td>103</td>
<td>868739278292</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>357</td>
<td>104</td>
<td>868739280425</td>
<td>104</td>
<td>868739280405</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>358</td>
<td>105</td>
<td>868739282529</td>
<td>105</td>
<td>868739282529</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>359</td>
<td>106</td>
<td>868739284672</td>
<td>106</td>
<td>868739284652</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>360</td>
<td>107</td>
<td>868739286835</td>
<td>107</td>
<td>868739286765</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>361</td>
<td>108</td>
<td>868739288968</td>
<td>108</td>
<td>868739288878</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>362</td>
<td>109</td>
<td>868739289911</td>
<td>109</td>
<td>8687392891001</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>363</td>
<td>110</td>
<td>868739293174</td>
<td>110</td>
<td>868739293114</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>364</td>
<td>111</td>
<td>868739295227</td>
<td>111</td>
<td>868739295227</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>365</td>
<td>112</td>
<td>868739297350</td>
<td>112</td>
<td>868739297350</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>366</td>
<td>113</td>
<td>868739299463</td>
<td>113</td>
<td>868739299463</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>367</td>
<td>114</td>
<td>868739301576</td>
<td>114</td>
<td>868739301576</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>368</td>
<td>115</td>
<td>868739303699</td>
<td>115</td>
<td>868739303699</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>369</td>
<td>116</td>
<td>868739305812</td>
<td>116</td>
<td>868739305812</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>370</td>
<td>117</td>
<td>868739307935</td>
<td>117</td>
<td>868739307935</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>371</td>
<td>118</td>
<td>868739310048</td>
<td>118</td>
<td>868739310048</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>372</td>
<td>119</td>
<td>868739312171</td>
<td>119</td>
<td>868739312171</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>373</td>
<td>120</td>
<td>868739314284</td>
<td>120</td>
<td>868739314284</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>374</td>
<td>121</td>
<td>868739316397</td>
<td>121</td>
<td>868739316397</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>375</td>
<td>122</td>
<td>868739318520</td>
<td>122</td>
<td>868739318520</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>376</td>
<td>123</td>
<td>868739320633</td>
<td>123</td>
<td>868739320633</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>377</td>
<td>124</td>
<td>868739322746</td>
<td>124</td>
<td>868739322746</td>
<td>151</td>
<td>75.5</td>
<td></td>
</tr>
<tr>
<td>378</td>
<td>125</td>
<td>868739324899</td>
<td>125</td>
<td>868739324899</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>379</td>
<td>0</td>
<td>868739327012</td>
<td>0</td>
<td>868739327012</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>380</td>
<td>1</td>
<td>868739329095</td>
<td>1</td>
<td>868739329095</td>
<td>31</td>
<td>15.5</td>
<td></td>
</tr>
<tr>
<td>381</td>
<td>2</td>
<td>868739331219</td>
<td>2</td>
<td>868739331219</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>382</td>
<td>3</td>
<td>868739333332</td>
<td>3</td>
<td>868739333332</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>383</td>
<td>4</td>
<td>868739335445</td>
<td>4</td>
<td>868739335445</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>384</td>
<td>5</td>
<td>868739337568</td>
<td>5</td>
<td>868739337568</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>385</td>
<td>6</td>
<td>868739339681</td>
<td>6</td>
<td>868739339681</td>
<td>120</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>386</td>
<td>7</td>
<td>868739341804</td>
<td>7</td>
<td>868739341804</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>387</td>
<td>8</td>
<td>868739343917</td>
<td>8</td>
<td>868739343917</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>388</td>
<td>9</td>
<td>868739346030</td>
<td>9</td>
<td>868739346030</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>389</td>
<td>10</td>
<td>868739348143</td>
<td>10</td>
<td>868739348143</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>390</td>
<td>11</td>
<td>868739350266</td>
<td>11</td>
<td>868739350266</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>391</td>
<td>12</td>
<td>868739352389</td>
<td>12</td>
<td>868739352389</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>392</td>
<td>13</td>
<td>868739354502</td>
<td>13</td>
<td>868739354502</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>393</td>
<td>14</td>
<td>868739356615</td>
<td>14</td>
<td>868739356615</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>394</td>
<td>15</td>
<td>868739358738</td>
<td>15</td>
<td>868739358738</td>
<td>150</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>395</td>
<td>16</td>
<td>868739360851</td>
<td>16</td>
<td>868739360851</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>396</td>
<td>17</td>
<td>868739362964</td>
<td>17</td>
<td>868739362964</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>397</td>
<td>18</td>
<td>868739365087</td>
<td>18</td>
<td>868739365087</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>398</td>
<td>19</td>
<td>868739367200</td>
<td>19</td>
<td>868739367200</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>399</td>
<td>20</td>
<td>868739369323</td>
<td>20</td>
<td>868739369323</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>21</td>
<td>868739371436</td>
<td>21</td>
<td>868739371436</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>401</td>
<td>22</td>
<td>868739373619</td>
<td>22</td>
<td>868739373549</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>402</td>
<td>23</td>
<td>868739373575</td>
<td>23</td>
<td>8687393735662</td>
<td>91</td>
<td>45.5</td>
<td></td>
</tr>
<tr>
<td>403</td>
<td>24</td>
<td>8687393737786</td>
<td>24</td>
<td>8687393737775</td>
<td>101</td>
<td>50.5</td>
<td></td>
</tr>
<tr>
<td>404</td>
<td>25</td>
<td>868739379989</td>
<td>25</td>
<td>8687393799899</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>405</td>
<td>26</td>
<td>868739382112</td>
<td>26</td>
<td>8687393820222</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>406</td>
<td>27</td>
<td>868739384165</td>
<td>27</td>
<td>868739384135</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>407</td>
<td>28</td>
<td>868739386278</td>
<td>28</td>
<td>868739386248</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>408</td>
<td>29</td>
<td>868739388411</td>
<td>29</td>
<td>868739388371</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>409</td>
<td>30</td>
<td>868739390584</td>
<td>30</td>
<td>868739390484</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>410</td>
<td>31</td>
<td>868739392697</td>
<td>31</td>
<td>868739392597</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>411</td>
<td>32</td>
<td>868739394770</td>
<td>32</td>
<td>868739394720</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>412</td>
<td>33</td>
<td>868739396883</td>
<td>33</td>
<td>868739396833</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>413</td>
<td>34</td>
<td>868739399016</td>
<td>34</td>
<td>868739398946</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>414</td>
<td>35</td>
<td>868739401119</td>
<td>35</td>
<td>868739401069</td>
<td>50</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>415</td>
<td>36</td>
<td>868739403222</td>
<td>36</td>
<td>868739403182</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>416</td>
<td>37</td>
<td>868739405345</td>
<td>37</td>
<td>868739405305</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>417</td>
<td>38</td>
<td>868739407458</td>
<td>38</td>
<td>868739407418</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>418</td>
<td>39</td>
<td>868739409561</td>
<td>39</td>
<td>868739409531</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>419</td>
<td>40</td>
<td>868739411694</td>
<td>40</td>
<td>868739411654</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>420</td>
<td>41</td>
<td>868739413797</td>
<td>41</td>
<td>868739413767</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>421</td>
<td>42</td>
<td>868739415910</td>
<td>42</td>
<td>868739415890</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>422</td>
<td>43</td>
<td>868739418023</td>
<td>43</td>
<td>868739418003</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>423</td>
<td>44</td>
<td>868739420136</td>
<td>44</td>
<td>868739420116</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>424</td>
<td>45</td>
<td>8687394222345</td>
<td>45</td>
<td>868739422230</td>
<td>111</td>
<td>55.5</td>
<td></td>
</tr>
<tr>
<td>425</td>
<td>46</td>
<td>868739424372</td>
<td>46</td>
<td>868739424352</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>426</td>
<td>47</td>
<td>868739426486</td>
<td>47</td>
<td>868739426465</td>
<td>21</td>
<td>10.5</td>
<td></td>
</tr>
<tr>
<td>427</td>
<td>48</td>
<td>868739428699</td>
<td>48</td>
<td>868739428589</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>428</td>
<td>49</td>
<td>868739430812</td>
<td>49</td>
<td>868739430702</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>429</td>
<td>50</td>
<td>868739432915</td>
<td>50</td>
<td>868739432815</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>430</td>
<td>51</td>
<td>868739435028</td>
<td>51</td>
<td>868739434938</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>431</td>
<td>52</td>
<td>868739437091</td>
<td>52</td>
<td>868739437051</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>432</td>
<td>53</td>
<td>868739439204</td>
<td>53</td>
<td>868739439164</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>433</td>
<td>54</td>
<td>868739441317</td>
<td>54</td>
<td>868739441287</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>434</td>
<td>55</td>
<td>868739443430</td>
<td>55</td>
<td>868739443400</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>435</td>
<td>56</td>
<td>868739445543</td>
<td>56</td>
<td>868739445513</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>436</td>
<td>57</td>
<td>8687394476656</td>
<td>57</td>
<td>868739447636</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>437</td>
<td>58</td>
<td>868739449859</td>
<td>58</td>
<td>868739449749</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>438</td>
<td>59</td>
<td>868739451962</td>
<td>59</td>
<td>868739451872</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>439</td>
<td>60</td>
<td>868739454085</td>
<td>60</td>
<td>868739453985</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>440</td>
<td>61</td>
<td>868739456198</td>
<td>61</td>
<td>868739456108</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>441</td>
<td>62</td>
<td>868739458321</td>
<td>62</td>
<td>868739458221</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>442</td>
<td>63</td>
<td>868739460424</td>
<td>63</td>
<td>868739460334</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>443</td>
<td>64</td>
<td>868739462547</td>
<td>64</td>
<td>868739462457</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>444</td>
<td>65</td>
<td>868739464650</td>
<td>65</td>
<td>868739464570</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>445</td>
<td>66</td>
<td>868739466723</td>
<td>66</td>
<td>868739466683</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>446</td>
<td>67</td>
<td>868739468836</td>
<td>67</td>
<td>868739468806</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>447</td>
<td>68</td>
<td>868739470939</td>
<td>68</td>
<td>868739470919</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>448</td>
<td>69</td>
<td>868739473052</td>
<td>69</td>
<td>868739473032</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>449</td>
<td>70</td>
<td>868739475166</td>
<td>70</td>
<td>868739475155</td>
<td>11</td>
<td>5.5</td>
<td></td>
</tr>
<tr>
<td>450</td>
<td>71</td>
<td>868739477279</td>
<td>71</td>
<td>868739477269</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>451</td>
<td>72</td>
<td>868739479382</td>
<td>72</td>
<td>868739479382</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>452</td>
<td>73</td>
<td>868739481605</td>
<td>73</td>
<td>868739481505</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>----</td>
<td>---------------</td>
<td>----</td>
<td>---------------</td>
<td>-----</td>
<td>----</td>
<td></td>
</tr>
<tr>
<td>453</td>
<td>74</td>
<td>868739483718</td>
<td>74</td>
<td>868739483618</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>454</td>
<td>75</td>
<td>868739485821</td>
<td>75</td>
<td>868739485741</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>455</td>
<td>76</td>
<td>868739487944</td>
<td>76</td>
<td>868739487854</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>456</td>
<td>77</td>
<td>868739490067</td>
<td>77</td>
<td>868739489967</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>457</td>
<td>78</td>
<td>868739492110</td>
<td>78</td>
<td>868739492090</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>458</td>
<td>79</td>
<td>868739494223</td>
<td>79</td>
<td>868739494203</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>459</td>
<td>80</td>
<td>868739496396</td>
<td>80</td>
<td>868739496315</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>460</td>
<td>81</td>
<td>868739498519</td>
<td>81</td>
<td>868739498439</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>461</td>
<td>82</td>
<td>868739500642</td>
<td>82</td>
<td>868739500552</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>462</td>
<td>83</td>
<td>868739502755</td>
<td>83</td>
<td>868739502665</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>463</td>
<td>84</td>
<td>868739504858</td>
<td>84</td>
<td>868739504788</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>464</td>
<td>85</td>
<td>868739506981</td>
<td>85</td>
<td>868739506901</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>465</td>
<td>86</td>
<td>868739509114</td>
<td>86</td>
<td>868739509024</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>466</td>
<td>87</td>
<td>868739511237</td>
<td>87</td>
<td>868739511137</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>467</td>
<td>88</td>
<td>868739513350</td>
<td>88</td>
<td>868739513260</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>468</td>
<td>89</td>
<td>868739515453</td>
<td>89</td>
<td>868739515373</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>469</td>
<td>90</td>
<td>868739517566</td>
<td>90</td>
<td>868739517486</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>470</td>
<td>91</td>
<td>868739519710</td>
<td>91</td>
<td>868739519609</td>
<td>101</td>
<td>50.5</td>
<td></td>
</tr>
<tr>
<td>471</td>
<td>92</td>
<td>868739521833</td>
<td>92</td>
<td>868739521722</td>
<td>111</td>
<td>55.5</td>
<td></td>
</tr>
<tr>
<td>472</td>
<td>93</td>
<td>868739523956</td>
<td>93</td>
<td>868739523846</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>473</td>
<td>94</td>
<td>868739526069</td>
<td>94</td>
<td>868739525959</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>474</td>
<td>95</td>
<td>868739528182</td>
<td>95</td>
<td>868739528082</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>475</td>
<td>96</td>
<td>868739530295</td>
<td>96</td>
<td>868739530205</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>476</td>
<td>97</td>
<td>868739532418</td>
<td>97</td>
<td>868739532318</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>477</td>
<td>98</td>
<td>868739534531</td>
<td>98</td>
<td>868739534431</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>478</td>
<td>99</td>
<td>868739536654</td>
<td>99</td>
<td>868739536554</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>479</td>
<td>100</td>
<td>868739538772</td>
<td>100</td>
<td>868739538667</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>480</td>
<td>101</td>
<td>868739540840</td>
<td>101</td>
<td>868739540780</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>481</td>
<td>102</td>
<td>868739542973</td>
<td>102</td>
<td>868739542903</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>482</td>
<td>103</td>
<td>868739545086</td>
<td>103</td>
<td>868739545016</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>483</td>
<td>104</td>
<td>868739547209</td>
<td>104</td>
<td>868739547129</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>484</td>
<td>105</td>
<td>868739549312</td>
<td>105</td>
<td>868739549252</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>485</td>
<td>106</td>
<td>868739551405</td>
<td>106</td>
<td>868739551365</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>486</td>
<td>107</td>
<td>868739553518</td>
<td>107</td>
<td>868739553488</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>487</td>
<td>108</td>
<td>868739555671</td>
<td>108</td>
<td>868739555601</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>488</td>
<td>109</td>
<td>868739557784</td>
<td>109</td>
<td>868739557714</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>489</td>
<td>110</td>
<td>868739559907</td>
<td>110</td>
<td>868739559837</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>490</td>
<td>111</td>
<td>868739561970</td>
<td>111</td>
<td>868739561950</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>491</td>
<td>112</td>
<td>868739564083</td>
<td>112</td>
<td>868739564063</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>492</td>
<td>113</td>
<td>868739566206</td>
<td>113</td>
<td>868739566186</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>493</td>
<td>114</td>
<td>868739568319</td>
<td>114</td>
<td>868739568299</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>494</td>
<td>115</td>
<td>868739570523</td>
<td>115</td>
<td>868739570412</td>
<td>111</td>
<td>55.5</td>
<td></td>
</tr>
<tr>
<td>495</td>
<td>116</td>
<td>868739572556</td>
<td>116</td>
<td>868739572536</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>496</td>
<td>117</td>
<td>868739574669</td>
<td>117</td>
<td>868739574649</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>497</td>
<td>118</td>
<td>868739576792</td>
<td>118</td>
<td>868739576762</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>498</td>
<td>119</td>
<td>868739578905</td>
<td>119</td>
<td>868739578885</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>499</td>
<td>120</td>
<td>868739581078</td>
<td>120</td>
<td>868739580998</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>121</td>
<td>868739583181</td>
<td>121</td>
<td>868739583121</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>501</td>
<td>122</td>
<td>868739585304</td>
<td>122</td>
<td>868739585234</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>502</td>
<td>123</td>
<td>868739587427</td>
<td>123</td>
<td>868739587347</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>503</td>
<td>124</td>
<td>868739589540</td>
<td>124</td>
<td>868739589470</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>504</td>
<td>125</td>
<td>868739591613</td>
<td>125</td>
<td>868739591583</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>505</td>
<td>0</td>
<td>868739593716</td>
<td>0</td>
<td>868739593696</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>506</td>
<td>1</td>
<td>868739595839</td>
<td>1</td>
<td>868739595819</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>507</td>
<td>2</td>
<td>868739597942</td>
<td>2</td>
<td>868739597932</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>508</td>
<td>3</td>
<td>868739600075</td>
<td>3</td>
<td>868739600045</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>509</td>
<td>4</td>
<td>868739602198</td>
<td>4</td>
<td>868739602168</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>510</td>
<td>5</td>
<td>868739604311</td>
<td>5</td>
<td>868739604281</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>511</td>
<td>6</td>
<td>868739606434</td>
<td>6</td>
<td>868739606394</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>7</td>
<td>868739608587</td>
<td>7</td>
<td>868739608517</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>513</td>
<td>8</td>
<td>868739610710</td>
<td>8</td>
<td>868739610630</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>514</td>
<td>9</td>
<td>868739612823</td>
<td>9</td>
<td>868739612743</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>515</td>
<td>10</td>
<td>868739614946</td>
<td>10</td>
<td>868739614866</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>516</td>
<td>11</td>
<td>868739617050</td>
<td>11</td>
<td>868739616979</td>
<td>71</td>
<td>35.5</td>
<td></td>
</tr>
<tr>
<td>517</td>
<td>12</td>
<td>868739619173</td>
<td>12</td>
<td>868739619102</td>
<td>71</td>
<td>35.5</td>
<td></td>
</tr>
<tr>
<td>518</td>
<td>13</td>
<td>868739621306</td>
<td>13</td>
<td>868739621216</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>519</td>
<td>14</td>
<td>868739623439</td>
<td>14</td>
<td>868739623339</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>520</td>
<td>15</td>
<td>868739625552</td>
<td>15</td>
<td>868739625452</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>521</td>
<td>16</td>
<td>868739627655</td>
<td>16</td>
<td>868739627575</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>522</td>
<td>17</td>
<td>868739629758</td>
<td>17</td>
<td>868739629688</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>523</td>
<td>18</td>
<td>868739631871</td>
<td>18</td>
<td>868739631801</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>524</td>
<td>19</td>
<td>868739633994</td>
<td>19</td>
<td>868739633924</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>525</td>
<td>20</td>
<td>868739636107</td>
<td>20</td>
<td>868739636037</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>526</td>
<td>21</td>
<td>868739638280</td>
<td>21</td>
<td>868739638150</td>
<td>130</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>527</td>
<td>22</td>
<td>868739640293</td>
<td>22</td>
<td>868739640273</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>528</td>
<td>23</td>
<td>868739642406</td>
<td>23</td>
<td>868739642386</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>529</td>
<td>24</td>
<td>868739644609</td>
<td>24</td>
<td>868739644499</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>530</td>
<td>25</td>
<td>868739646732</td>
<td>25</td>
<td>868739646622</td>
<td>110</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>531</td>
<td>26</td>
<td>868739648755</td>
<td>26</td>
<td>868739648735</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>532</td>
<td>27</td>
<td>868739650958</td>
<td>27</td>
<td>868739650858</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>533</td>
<td>28</td>
<td>868739653071</td>
<td>28</td>
<td>868739652971</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>534</td>
<td>29</td>
<td>868739655194</td>
<td>29</td>
<td>868739655094</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>535</td>
<td>30</td>
<td>868739657297</td>
<td>30</td>
<td>868739657207</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>536</td>
<td>31</td>
<td>868739659410</td>
<td>31</td>
<td>868739659320</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>537</td>
<td>32</td>
<td>868739661533</td>
<td>32</td>
<td>868739661443</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>538</td>
<td>33</td>
<td>868739663586</td>
<td>33</td>
<td>868739663556</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>539</td>
<td>34</td>
<td>868739665689</td>
<td>34</td>
<td>868739665669</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>540</td>
<td>35</td>
<td>868739667783</td>
<td>35</td>
<td>868739667792</td>
<td>81</td>
<td>40.5</td>
<td></td>
</tr>
<tr>
<td>541</td>
<td>36</td>
<td>868739670717</td>
<td>36</td>
<td>868739669906</td>
<td>81</td>
<td>40.5</td>
<td></td>
</tr>
<tr>
<td>542</td>
<td>37</td>
<td>868739672089</td>
<td>37</td>
<td>868739672019</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>543</td>
<td>38</td>
<td>868739674172</td>
<td>38</td>
<td>868739674142</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>544</td>
<td>39</td>
<td>868739676375</td>
<td>39</td>
<td>868739676255</td>
<td>120</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>545</td>
<td>40</td>
<td>868739678468</td>
<td>40</td>
<td>868739678368</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>546</td>
<td>41</td>
<td>868739680581</td>
<td>41</td>
<td>868739680491</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>547</td>
<td>42</td>
<td>868739682724</td>
<td>42</td>
<td>868739682604</td>
<td>120</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>548</td>
<td>43</td>
<td>868739684757</td>
<td>43</td>
<td>868739684727</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>549</td>
<td>44</td>
<td>868739686870</td>
<td>44</td>
<td>868739686840</td>
<td>30</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>550</td>
<td>45</td>
<td>868739688993</td>
<td>45</td>
<td>868739688953</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>551</td>
<td>46</td>
<td>868739691146</td>
<td>46</td>
<td>868739691076</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>552</td>
<td>47</td>
<td>868739693209</td>
<td>47</td>
<td>868739693189</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>553</td>
<td>48</td>
<td>868739695392</td>
<td>48</td>
<td>868739695302</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
</tbody>
</table>

187
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>554</td>
<td>49</td>
<td>868739697535</td>
<td>49</td>
<td>868739697425</td>
<td>110</td>
</tr>
<tr>
<td>555</td>
<td>50</td>
<td>868739699578</td>
<td>50</td>
<td>868739699538</td>
<td>40</td>
</tr>
<tr>
<td>556</td>
<td>51</td>
<td>868739701791</td>
<td>51</td>
<td>868739701661</td>
<td>130</td>
</tr>
<tr>
<td>557</td>
<td>52</td>
<td>868739703844</td>
<td>52</td>
<td>868739703774</td>
<td>70</td>
</tr>
<tr>
<td>558</td>
<td>53</td>
<td>868739705917</td>
<td>53</td>
<td>868739705887</td>
<td>30</td>
</tr>
<tr>
<td>559</td>
<td>54</td>
<td>868739708130</td>
<td>54</td>
<td>868739708010</td>
<td>120</td>
</tr>
<tr>
<td>560</td>
<td>55</td>
<td>868739710193</td>
<td>55</td>
<td>868739710123</td>
<td>70</td>
</tr>
<tr>
<td>561</td>
<td>56</td>
<td>868739712367</td>
<td>56</td>
<td>868739712246</td>
<td>121</td>
</tr>
<tr>
<td>562</td>
<td>57</td>
<td>868739714500</td>
<td>57</td>
<td>868739714359</td>
<td>141</td>
</tr>
<tr>
<td>563</td>
<td>58</td>
<td>868739716553</td>
<td>58</td>
<td>868739716483</td>
<td>70</td>
</tr>
<tr>
<td>564</td>
<td>59</td>
<td>868739718636</td>
<td>59</td>
<td>868739718596</td>
<td>40</td>
</tr>
<tr>
<td>565</td>
<td>60</td>
<td>868739720839</td>
<td>60</td>
<td>868739720709</td>
<td>130</td>
</tr>
<tr>
<td>566</td>
<td>61</td>
<td>868739722922</td>
<td>61</td>
<td>868739722832</td>
<td>90</td>
</tr>
<tr>
<td>567</td>
<td>62</td>
<td>868739725025</td>
<td>62</td>
<td>868739724945</td>
<td>80</td>
</tr>
<tr>
<td>568</td>
<td>63</td>
<td>868739727088</td>
<td>63</td>
<td>868739727058</td>
<td>30</td>
</tr>
<tr>
<td>569</td>
<td>64</td>
<td>868739729251</td>
<td>64</td>
<td>868739729181</td>
<td>70</td>
</tr>
<tr>
<td>570</td>
<td>65</td>
<td>868739731384</td>
<td>65</td>
<td>868739731294</td>
<td>90</td>
</tr>
<tr>
<td>571</td>
<td>66</td>
<td>868739733487</td>
<td>66</td>
<td>868739733407</td>
<td>80</td>
</tr>
<tr>
<td>572</td>
<td>67</td>
<td>868739735550</td>
<td>67</td>
<td>868739735530</td>
<td>20</td>
</tr>
<tr>
<td>573</td>
<td>68</td>
<td>868739737743</td>
<td>68</td>
<td>868739737643</td>
<td>100</td>
</tr>
<tr>
<td>574</td>
<td>69</td>
<td>868739739846</td>
<td>69</td>
<td>868739739756</td>
<td>90</td>
</tr>
<tr>
<td>575</td>
<td>70</td>
<td>868739741929</td>
<td>70</td>
<td>868739741879</td>
<td>50</td>
</tr>
<tr>
<td>576</td>
<td>71</td>
<td>868739744042</td>
<td>71</td>
<td>868739743992</td>
<td>50</td>
</tr>
<tr>
<td>577</td>
<td>72</td>
<td>868739746145</td>
<td>72</td>
<td>868739746105</td>
<td>40</td>
</tr>
<tr>
<td>578</td>
<td>73</td>
<td>868739748318</td>
<td>73</td>
<td>868739748228</td>
<td>90</td>
</tr>
<tr>
<td>579</td>
<td>74</td>
<td>868739750421</td>
<td>74</td>
<td>868739750341</td>
<td>80</td>
</tr>
<tr>
<td>580</td>
<td>75</td>
<td>868739752534</td>
<td>75</td>
<td>868739752454</td>
<td>80</td>
</tr>
<tr>
<td>581</td>
<td>76</td>
<td>868739754597</td>
<td>76</td>
<td>868739754577</td>
<td>20</td>
</tr>
<tr>
<td>582</td>
<td>77</td>
<td>868739756710</td>
<td>77</td>
<td>868739756690</td>
<td>20</td>
</tr>
<tr>
<td>583</td>
<td>78</td>
<td>868739758944</td>
<td>78</td>
<td>868739758803</td>
<td>141</td>
</tr>
<tr>
<td>584</td>
<td>79</td>
<td>868739761027</td>
<td>79</td>
<td>868739760926</td>
<td>101</td>
</tr>
<tr>
<td>585</td>
<td>80</td>
<td>868739763090</td>
<td>80</td>
<td>868739763039</td>
<td>51</td>
</tr>
<tr>
<td>586</td>
<td>81</td>
<td>868739765243</td>
<td>81</td>
<td>868739765162</td>
<td>81</td>
</tr>
<tr>
<td>587</td>
<td>82</td>
<td>868739767376</td>
<td>82</td>
<td>868739767276</td>
<td>100</td>
</tr>
<tr>
<td>588</td>
<td>83</td>
<td>868739769519</td>
<td>83</td>
<td>868739769389</td>
<td>130</td>
</tr>
<tr>
<td>589</td>
<td>84</td>
<td>868739771662</td>
<td>84</td>
<td>868739771542</td>
<td>120</td>
</tr>
<tr>
<td>590</td>
<td>85</td>
<td>868739773785</td>
<td>85</td>
<td>868739773655</td>
<td>130</td>
</tr>
<tr>
<td>591</td>
<td>86</td>
<td>868739775868</td>
<td>86</td>
<td>868739775768</td>
<td>100</td>
</tr>
<tr>
<td>592</td>
<td>87</td>
<td>868739777941</td>
<td>87</td>
<td>868739777891</td>
<td>50</td>
</tr>
<tr>
<td>593</td>
<td>88</td>
<td>868739780064</td>
<td>88</td>
<td>868739780004</td>
<td>60</td>
</tr>
<tr>
<td>594</td>
<td>89</td>
<td>868739782227</td>
<td>89</td>
<td>868739782117</td>
<td>110</td>
</tr>
<tr>
<td>595</td>
<td>90</td>
<td>868739784360</td>
<td>90</td>
<td>868739784240</td>
<td>120</td>
</tr>
<tr>
<td>596</td>
<td>91</td>
<td>868739786413</td>
<td>91</td>
<td>868739786353</td>
<td>60</td>
</tr>
<tr>
<td>597</td>
<td>92</td>
<td>868739788516</td>
<td>92</td>
<td>868739788466</td>
<td>50</td>
</tr>
<tr>
<td>598</td>
<td>93</td>
<td>868739790719</td>
<td>93</td>
<td>868739790589</td>
<td>130</td>
</tr>
<tr>
<td>599</td>
<td>94</td>
<td>868739792802</td>
<td>94</td>
<td>868739792712</td>
<td>90</td>
</tr>
<tr>
<td>600</td>
<td>95</td>
<td>868739794825</td>
<td>95</td>
<td>868739794825</td>
<td>180</td>
</tr>
<tr>
<td>601</td>
<td>96</td>
<td>868739796938</td>
<td>96</td>
<td>868739796938</td>
<td>120</td>
</tr>
<tr>
<td>602</td>
<td>97</td>
<td>868739799061</td>
<td>97</td>
<td>868739799061</td>
<td>30</td>
</tr>
<tr>
<td>603</td>
<td>98</td>
<td>868739801264</td>
<td>98</td>
<td>868739801184</td>
<td>80</td>
</tr>
<tr>
<td>604</td>
<td>99</td>
<td>868739803317</td>
<td>99</td>
<td>868739803297</td>
<td>20</td>
</tr>
</tbody>
</table>

188
### APPENDIX R. EXPERIMENT 2 DATA FOR UDP

**Experiment 2 (Server Control using UDP, Windows NT and Internet Explorer)**

<table>
<thead>
<tr>
<th>N</th>
<th>stat_u.txt</th>
<th>stat_u.txt</th>
<th>sent_u.txt</th>
<th>sent_u.txt</th>
<th>Round Trip Latency Time</th>
<th>Calculated one way Latency time</th>
<th>Freq. Bin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>868736297196</td>
<td>0</td>
<td>868736297076</td>
<td>120</td>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>868736299239</td>
<td>1</td>
<td>868736299199</td>
<td>40</td>
<td>20</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>868736301342</td>
<td>2</td>
<td>868736301312</td>
<td>30</td>
<td>15</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>868736303465</td>
<td>3</td>
<td>868736303425</td>
<td>40</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>868736305568</td>
<td>4</td>
<td>868736305548</td>
<td>20</td>
<td>10</td>
<td>21</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>868736307681</td>
<td>5</td>
<td>868736307661</td>
<td>20</td>
<td>10</td>
<td>26</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>868736309814</td>
<td>6</td>
<td>868736309774</td>
<td>40</td>
<td>20</td>
<td>31</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>868736311917</td>
<td>7</td>
<td>868736311897</td>
<td>20</td>
<td>10</td>
<td>36</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>868736314030</td>
<td>8</td>
<td>868736314010</td>
<td>20</td>
<td>10</td>
<td>41</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>868736316153</td>
<td>9</td>
<td>868736316123</td>
<td>30</td>
<td>15</td>
<td>46</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>868736318266</td>
<td>10</td>
<td>868736318246</td>
<td>20</td>
<td>10</td>
<td>51</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>868736320379</td>
<td>11</td>
<td>868736320359</td>
<td>20</td>
<td>10</td>
<td>56</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
<td>868736322492</td>
<td>12</td>
<td>868736322472</td>
<td>20</td>
<td>10</td>
<td>61</td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>868736324615</td>
<td>13</td>
<td>868736324595</td>
<td>20</td>
<td>10</td>
<td>66</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>868736326728</td>
<td>14</td>
<td>868736326708</td>
<td>20</td>
<td>10</td>
<td>71</td>
</tr>
<tr>
<td>16</td>
<td>15</td>
<td>868736328851</td>
<td>15</td>
<td>868736328821</td>
<td>30</td>
<td>15</td>
<td>76</td>
</tr>
<tr>
<td>17</td>
<td>16</td>
<td>868736330954</td>
<td>16</td>
<td>868736330944</td>
<td>10</td>
<td>5</td>
<td>81</td>
</tr>
<tr>
<td>18</td>
<td>17</td>
<td>868736333077</td>
<td>17</td>
<td>868736333057</td>
<td>20</td>
<td>10</td>
<td>86</td>
</tr>
<tr>
<td>19</td>
<td>18</td>
<td>868736335281</td>
<td>18</td>
<td>868736335170</td>
<td>111</td>
<td>55.5</td>
<td>91</td>
</tr>
<tr>
<td>20</td>
<td>19</td>
<td>868736337384</td>
<td>19</td>
<td>868736337293</td>
<td>91</td>
<td>45.5</td>
<td>96</td>
</tr>
<tr>
<td>21</td>
<td>20</td>
<td>868736339497</td>
<td>20</td>
<td>868736339407</td>
<td>90</td>
<td>45</td>
<td>101</td>
</tr>
<tr>
<td>22</td>
<td>21</td>
<td>868736341610</td>
<td>21</td>
<td>868736341530</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>22</td>
<td>868736343723</td>
<td>22</td>
<td>868736343643</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>23</td>
<td>868736345836</td>
<td>23</td>
<td>868736345766</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>24</td>
<td>868736347969</td>
<td>24</td>
<td>868736347879</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>25</td>
<td>868736350072</td>
<td>25</td>
<td>868736349992</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>26</td>
<td>868736352175</td>
<td>26</td>
<td>868736352115</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>27</td>
<td>868736354288</td>
<td>27</td>
<td>868736354228</td>
<td>60</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>28</td>
<td>868736356381</td>
<td>28</td>
<td>868736356341</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>29</td>
<td>868736358484</td>
<td>29</td>
<td>868736358464</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>868736360597</td>
<td>30</td>
<td>868736360577</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>31</td>
<td>868736362780</td>
<td>31</td>
<td>868736362690</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>32</td>
<td>868736364893</td>
<td>32</td>
<td>868736364813</td>
<td>80</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>33</td>
<td>868736366996</td>
<td>33</td>
<td>868736366926</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>34</td>
<td>868736369109</td>
<td>34</td>
<td>868736369039</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>35</td>
<td>868736371232</td>
<td>35</td>
<td>868736371162</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>36</td>
<td>868736373375</td>
<td>36</td>
<td>868736373275</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>37</td>
<td>868736375488</td>
<td>37</td>
<td>868736375398</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>38</td>
<td>868736377601</td>
<td>38</td>
<td>868736377511</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>39</td>
<td>868736379704</td>
<td>39</td>
<td>868736379834</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>40</td>
<td>868736381817</td>
<td>40</td>
<td>868736381747</td>
<td>70</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>41</td>
<td>868736383961</td>
<td>41</td>
<td>868736383860</td>
<td>101</td>
<td>50.5</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>42</td>
<td>868736386014</td>
<td>42</td>
<td>868736385983</td>
<td>31</td>
<td>15.5</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>43</td>
<td>868736388187</td>
<td>43</td>
<td>868736388097</td>
<td>90</td>
<td>45</td>
<td></td>
</tr>
</tbody>
</table>
|   |   | 868736390300 |   | 868736390210 | 90 | 45
|---|---|-------------|---|-------------|----|---
| 45| 44| 868736392413 | 45| 868736392333 | 80 | 40
| 46| 46| 868736394516 | 46| 868736394456 | 60 | 30
| 47| 47| 868736396639 | 47| 868736396569 | 70 | 35
| 48| 48| 868736398752 | 48| 868736398692 | 60 | 30
| 49| 49| 868736400865 | 49| 868736400805 | 60 | 30
| 50| 50| 868736402968 | 50| 868736402918 | 50 | 25
| 51| 51| 868736405071 | 51| 868736405041 | 30 | 15
| 52| 52| 868736407174 | 52| 868736407154 | 20 | 10
| 53| 53| 868736409337 | 53| 868736409267 | 70 | 35
| 54| 54| 868736411450 | 54| 868736411390 | 60 | 30
| 55| 55| 868736413553 | 55| 868736413503 | 50 | 25
| 56| 56| 868736415666 | 56| 868736415616 | 50 | 25
| 57| 57| 868736417779 | 57| 868736417739 | 40 | 20
| 58| 58| 868736419902 | 58| 868736419852 | 50 | 25
| 59| 59| 868736422075 | 59| 868736421975 | 100| 50
| 60| 60| 868736424198 | 60| 868736424088 | 110| 55
| 61| 61| 868736426311 | 61| 868736426211 | 100| 50
| 62| 62| 868736428414 | 62| 868736428324 | 90 | 45
| 63| 63| 868736430528 | 63| 868736430437 | 91 | 45.5
| 64| 64| 868736432631 | 64| 868736432560 | 71 | 35.5
| 65| 65| 868736434734 | 65| 868736434673 | 61 | 30.5
| 66| 66| 868736436837 | 66| 868736436797 | 40 | 20
| 67| 67| 868736438940 | 67| 868736438910 | 30 | 15
| 68| 68| 868736441043 | 68| 868736441023 | 20 | 10
| 69| 69| 868736443246 | 69| 868736443146 | 100| 50
| 70| 70| 868736445279 | 70| 868736445259 | 20 | 10
| 71| 71| 868736447392 | 71| 868736447372 | 20 | 10
| 72| 72| 868736449515 | 72| 868736449495 | 20 | 10
| 73| 73| 868736451718 | 73| 868736451608 | 110| 55
| 74| 74| 868736453821 | 74| 868736453731 | 90 | 45
| 75| 75| 868736455924 | 75| 868736455844 | 80 | 40
| 76| 76| 868736458047 | 76| 868736457967 | 80 | 40
| 77| 77| 868736460160 | 77| 868736460080 | 80 | 40
| 78| 78| 868736462273 | 78| 868736462193 | 80 | 40
| 79| 79| 868736464376 | 79| 868736464316 | 60 | 30
| 80| 80| 868736466489 | 80| 868736466429 | 60 | 30
| 81| 81| 868736468602 | 81| 868736468542 | 60 | 30
| 82| 82| 868736470705 | 82| 868736470665 | 40 | 20
| 83| 83| 868736472808 | 83| 868736472778 | 30 | 15
| 84| 84| 868736474911 | 84| 868736474901 | 10 | 5
| 85| 85| 868736477125 | 85| 868736477014 | 111| 55.5
| 86| 86| 868736479228 | 86| 868736479137 | 91 | 45.5
| 87| 87| 868736481341 | 87| 868736481250 | 91 | 45.5
| 88| 88| 868736483444 | 88| 868736483374 | 70 | 35
| 89| 89| 868736485557 | 89| 868736485487 | 70 | 35
| 90| 90| 868736487630 | 90| 868736487600 | 30 | 15
| 91| 91| 868736489733 | 91| 868736489723 | 10 | 5
| 92| 92| 868736491856 | 92| 868736491836 | 20 | 10
| 93| 93| 868736493905 | 93| 868736493949 | 110| 55
| 94| 94| 868736496172 | 94| 868736496072 | 100| 50

192
<p>| 96 | 95 | 868736498295 | 95 | 868736498185 | 110 | 55 |
| 97 | 96 | 868736500338 | 96 | 868736500308 | 30  | 15 |
| 98 | 97 | 868736502441 | 97 | 868736502421 | 20  | 10 |
| 99 | 98 | 868736504604 | 98 | 868736504534 | 70  | 35 |
|100 | 99 | 868736506717 | 99 | 868736506657 | 60  | 30 |
|101 |100 | 868736508840 |100 | 868736508770 | 70  | 35 |
|102 |101 | 868736510913 |101 | 868736510883 | 30  | 15 |
|103 |102 | 868736513026 |102 | 868736513006 | 20  | 10 |
|104 |103 | 868736515139 |103 | 868736515119 | 20  | 10 |
|105 |104 | 868736517252 |104 | 868736517242 | 10  |  5 |
|106 |105 | 868736519455 |105 | 868736519355 |100 | 50 |
|107 |106 | 868736521498 |106 | 868736521468 | 30  | 15 |
|108 |107 | 868736523651 |107 | 868736523591 | 60  | 30 |
|109 |108 | 868736525744 |108 | 868736525704 | 40  | 20 |
|110 |109 | 868736527847 |109 | 868736527817 | 30  | 15 |
|111 |110 | 868736529950 |110 | 868736529940 | 10  |  5 |
|112 |111 | 868736531254 |111 | 868736532054 |100 | 50 |
|113 |112 | 868736534257 |112 | 868736534177 | 80  | 40 |
|114 |113 | 868736536380 |113 | 868736536290 | 90  | 45 |
|115 |114 | 868736538493 |114 | 868736538403 | 90  | 45 |
|116 |115 | 868736540586 |115 | 868736540526 | 60  | 30 |
|117 |116 | 868736542699 |116 | 868736542639 | 60  | 30 |
|118 |117 | 868736544802 |117 | 868736544752 | 50  | 25 |
|119 |118 | 868736546915 |118 | 868736546875 | 40  | 20 |
|120 |119 | 868736549018 |119 | 868736548988 | 30  | 15 |
|121 |120 | 868736551141 |120 | 868736551101 | 40  | 20 |
|122 |121 | 868736553234 |121 | 868736553224 | 10  |  5 |
|123 |122 | 868736555357 |122 | 868736555337 | 20  | 10 |
|124 |123 | 868736557480 |123 | 868736557460 | 20  | 10 |
|125 |124 | 868736559603 |124 | 868736559573 | 30  | 15 |
|126 |125 | 868736561716 |125 | 868736561686 | 30  | 15 |
|127 |  0 | 868736563829 |  0 | 868736563809 | 20  | 10 |
|128 |  1 | 868736566032 |  1 | 868736565922 |110 | 55 |
|129 |  2 | 868736568145 |  2 | 868736568045 |100 | 50 |
|130 |  3 | 868736570248 |  3 | 868736570158 | 90  | 45 |
|131 |  4 | 868736572361 |  4 | 868736572271 | 90  | 45 |
|132 |  5 | 868736574495 |  5 | 868736574394 |101 | 50.5|
|133 |  6 | 868736576558 |  6 | 868736576507 | 51  | 25.5|
|134 |  7 | 868736578661 |  7 | 868736578630 | 31  | 15.5|
|135 |  8 | 868736580794 |  8 | 868736580744 | 50  | 25 |
|136 |  9 | 868736582897 |  9 | 868736582857 | 40  | 20 |
|137 | 10 | 868736585010 | 10 | 868736584980 | 30  | 15 |
|138 | 11 | 868736587123 | 11 | 868736587093 | 30  | 15 |
|139 | 12 | 868736589226 | 12 | 868736589206 | 20  | 10 |
|140 | 13 | 868736591349 | 13 | 868736591329 | 20  | 10 |
|141 | 14 | 868736593462 | 14 | 868736593442 | 20  | 10 |
|142 | 15 | 868736595565 | 15 | 868736595555 |110 | 55 |
|143 | 16 | 868736597788 | 16 | 868736597768 |110 | 55 |
|144 | 17 | 868736599951 | 17 | 868736599791 |160 | 80 |
|145 | 18 | 868736601974 | 18 | 868736601914 | 60  | 30 |
|146 | 19 | 868736604107 | 19 | 868736604027 | 80  | 40 |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>147</td>
<td>20</td>
<td>868736606220</td>
<td>20</td>
<td>868736606140</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>148</td>
<td>21</td>
<td>868736608323</td>
<td>21</td>
<td>868736608263</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>149</td>
<td>22</td>
<td>868736610446</td>
<td>22</td>
<td>868736610376</td>
<td>70</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>23</td>
<td>868736612549</td>
<td>23</td>
<td>868736612489</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>151</td>
<td>24</td>
<td>868736614662</td>
<td>24</td>
<td>868736614612</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>152</td>
<td>25</td>
<td>868736616765</td>
<td>25</td>
<td>868736616725</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>153</td>
<td>26</td>
<td>868736618838</td>
<td>26</td>
<td>868736618848</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>154</td>
<td>27</td>
<td>868736620962</td>
<td>27</td>
<td>868736620961</td>
<td>101</td>
<td>50.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>155</td>
<td>28</td>
<td>868736623165</td>
<td>28</td>
<td>868736623074</td>
<td>91</td>
<td>45.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>156</td>
<td>29</td>
<td>868736625268</td>
<td>29</td>
<td>868736625197</td>
<td>71</td>
<td>35.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>157</td>
<td>30</td>
<td>868736627391</td>
<td>30</td>
<td>868736627310</td>
<td>81</td>
<td>40.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>158</td>
<td>31</td>
<td>868736629564</td>
<td>31</td>
<td>868736629434</td>
<td>130</td>
<td>65</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>159</td>
<td>32</td>
<td>868736631587</td>
<td>32</td>
<td>868736631547</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>33</td>
<td>868736633700</td>
<td>33</td>
<td>868736633660</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>161</td>
<td>34</td>
<td>868736635813</td>
<td>34</td>
<td>868736635783</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>162</td>
<td>35</td>
<td>868736637936</td>
<td>35</td>
<td>868736637896</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>163</td>
<td>36</td>
<td>868736640049</td>
<td>36</td>
<td>868736640009</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>164</td>
<td>37</td>
<td>868736642152</td>
<td>37</td>
<td>868736642132</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>165</td>
<td>38</td>
<td>868736644275</td>
<td>38</td>
<td>868736644245</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>166</td>
<td>39</td>
<td>868736646388</td>
<td>39</td>
<td>868736646358</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>167</td>
<td>40</td>
<td>868736648501</td>
<td>40</td>
<td>868736648481</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>168</td>
<td>41</td>
<td>868736650624</td>
<td>41</td>
<td>868736650604</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>169</td>
<td>42</td>
<td>868736652747</td>
<td>42</td>
<td>868736652717</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>170</td>
<td>43</td>
<td>868736654850</td>
<td>43</td>
<td>868736654830</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>171</td>
<td>44</td>
<td>868736656963</td>
<td>44</td>
<td>868736656953</td>
<td>110</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>172</td>
<td>45</td>
<td>868736659166</td>
<td>45</td>
<td>868736659066</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>173</td>
<td>46</td>
<td>868736661269</td>
<td>46</td>
<td>868736661189</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>47</td>
<td>868736663402</td>
<td>47</td>
<td>868736663302</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>175</td>
<td>48</td>
<td>868736665445</td>
<td>48</td>
<td>868736665415</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>176</td>
<td>49</td>
<td>868736667568</td>
<td>49</td>
<td>868736667538</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>177</td>
<td>50</td>
<td>868736669681</td>
<td>50</td>
<td>868736669651</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>178</td>
<td>51</td>
<td>868736671794</td>
<td>51</td>
<td>868736671764</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>179</td>
<td>52</td>
<td>868736673897</td>
<td>52</td>
<td>868736673887</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>180</td>
<td>53</td>
<td>868736676031</td>
<td>53</td>
<td>868736676001</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>181</td>
<td>54</td>
<td>868736678144</td>
<td>54</td>
<td>868736678114</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>182</td>
<td>55</td>
<td>868736680347</td>
<td>55</td>
<td>868736680237</td>
<td>110</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>183</td>
<td>56</td>
<td>868736682570</td>
<td>56</td>
<td>868736682530</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>184</td>
<td>57</td>
<td>868736684743</td>
<td>57</td>
<td>868736684733</td>
<td>10</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>185</td>
<td>58</td>
<td>868736686636</td>
<td>58</td>
<td>868736686586</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>186</td>
<td>59</td>
<td>868736688729</td>
<td>59</td>
<td>868736688699</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>187</td>
<td>60</td>
<td>868736690872</td>
<td>60</td>
<td>868736690822</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>188</td>
<td>61</td>
<td>868736692985</td>
<td>61</td>
<td>868736692935</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>189</td>
<td>62</td>
<td>868736695098</td>
<td>62</td>
<td>868736695048</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>190</td>
<td>63</td>
<td>868736697217</td>
<td>63</td>
<td>868736697171</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>191</td>
<td>64</td>
<td>868736699394</td>
<td>64</td>
<td>868736699294</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>192</td>
<td>65</td>
<td>868736701567</td>
<td>65</td>
<td>868736701407</td>
<td>160</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>193</td>
<td>66</td>
<td>868736703570</td>
<td>66</td>
<td>868736703520</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>194</td>
<td>67</td>
<td>868736705693</td>
<td>67</td>
<td>868736705643</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>195</td>
<td>68</td>
<td>868736707846</td>
<td>68</td>
<td>868736707756</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>196</td>
<td>69</td>
<td>868736709949</td>
<td>69</td>
<td>868736709879</td>
<td>70</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>197</td>
<td>70</td>
<td>868736712062</td>
<td>70</td>
<td>868736711992</td>
<td>70</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>198</td>
<td>71</td>
<td>868736714165</td>
<td>71</td>
<td>868736714105</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>199</td>
<td>72</td>
<td>868736716228</td>
<td>72</td>
<td>868736716228</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>73</td>
<td>868736718391</td>
<td>73</td>
<td>868736718341</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>201</td>
<td>74</td>
<td>868736720484</td>
<td>74</td>
<td>868736720454</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>202</td>
<td>75</td>
<td>868736722598</td>
<td>75</td>
<td>868736722577</td>
<td>21</td>
<td>10.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>203</td>
<td>76</td>
<td>868736724711</td>
<td>76</td>
<td>868736724691</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>204</td>
<td>77</td>
<td>868736726914</td>
<td>77</td>
<td>868736726804</td>
<td>110</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>205</td>
<td>78</td>
<td>868736729017</td>
<td>78</td>
<td>868736728927</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>206</td>
<td>79</td>
<td>868736731140</td>
<td>79</td>
<td>868736731040</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>207</td>
<td>80</td>
<td>868736733243</td>
<td>80</td>
<td>868736733163</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>208</td>
<td>81</td>
<td>868736735366</td>
<td>81</td>
<td>868736735276</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>209</td>
<td>82</td>
<td>868736737419</td>
<td>82</td>
<td>868736737389</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>210</td>
<td>83</td>
<td>868736739532</td>
<td>83</td>
<td>868736739512</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>211</td>
<td>84</td>
<td>868736741645</td>
<td>84</td>
<td>868736741625</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>212</td>
<td>85</td>
<td>868736743768</td>
<td>85</td>
<td>868736743738</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>213</td>
<td>86</td>
<td>868736745871</td>
<td>86</td>
<td>868736745861</td>
<td>10</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>214</td>
<td>87</td>
<td>868736747994</td>
<td>87</td>
<td>868736747974</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>215</td>
<td>88</td>
<td>868736750137</td>
<td>88</td>
<td>868736750087</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>216</td>
<td>89</td>
<td>868736752250</td>
<td>89</td>
<td>868736752210</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>217</td>
<td>90</td>
<td>868736754373</td>
<td>90</td>
<td>868736754323</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>218</td>
<td>91</td>
<td>868736756476</td>
<td>91</td>
<td>868736756446</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>219</td>
<td>92</td>
<td>868736758589</td>
<td>92</td>
<td>868736758559</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>220</td>
<td>93</td>
<td>868736760712</td>
<td>93</td>
<td>868736760672</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>221</td>
<td>94</td>
<td>868736762815</td>
<td>94</td>
<td>868736762795</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>222</td>
<td>95</td>
<td>868736764928</td>
<td>95</td>
<td>868736764908</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>223</td>
<td>96</td>
<td>868736767081</td>
<td>96</td>
<td>868736767021</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>224</td>
<td>97</td>
<td>868736769195</td>
<td>97</td>
<td>868736769144</td>
<td>51</td>
<td>25.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>225</td>
<td>98</td>
<td>868736771328</td>
<td>98</td>
<td>868736771257</td>
<td>71</td>
<td>35.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>226</td>
<td>99</td>
<td>868736773431</td>
<td>99</td>
<td>868736773371</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>227</td>
<td>100</td>
<td>868736775544</td>
<td>100</td>
<td>868736775494</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>228</td>
<td>101</td>
<td>868736777667</td>
<td>101</td>
<td>868736777607</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>229</td>
<td>102</td>
<td>868736779760</td>
<td>102</td>
<td>868736779720</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>230</td>
<td>103</td>
<td>868736781883</td>
<td>103</td>
<td>868736781843</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>231</td>
<td>104</td>
<td>868736783996</td>
<td>104</td>
<td>868736783956</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>232</td>
<td>105</td>
<td>868736786099</td>
<td>105</td>
<td>868736786079</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>233</td>
<td>106</td>
<td>868736788212</td>
<td>106</td>
<td>868736788192</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>234</td>
<td>107</td>
<td>868736790325</td>
<td>107</td>
<td>868736790305</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>235</td>
<td>108</td>
<td>868736792438</td>
<td>108</td>
<td>868736792428</td>
<td>10</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>236</td>
<td>109</td>
<td>868736794561</td>
<td>109</td>
<td>868736794541</td>
<td>110</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>237</td>
<td>110</td>
<td>868736796754</td>
<td>110</td>
<td>868736796654</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>238</td>
<td>111</td>
<td>868736798857</td>
<td>111</td>
<td>868736798777</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>239</td>
<td>112</td>
<td>868736800970</td>
<td>112</td>
<td>868736800890</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>240</td>
<td>113</td>
<td>868736803083</td>
<td>113</td>
<td>868736803013</td>
<td>70</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>241</td>
<td>114</td>
<td>868736805206</td>
<td>114</td>
<td>868736805126</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>242</td>
<td>115</td>
<td>868736807339</td>
<td>115</td>
<td>868736807239</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>243</td>
<td>116</td>
<td>868736809462</td>
<td>116</td>
<td>868736809362</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>244</td>
<td>117</td>
<td>868736811685</td>
<td>117</td>
<td>868736811475</td>
<td>110</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>245</td>
<td>118</td>
<td>868736813709</td>
<td>118</td>
<td>868736813598</td>
<td>111</td>
<td>55.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>246</td>
<td>119</td>
<td>868736815751</td>
<td>119</td>
<td>868736815721</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>247</td>
<td>120</td>
<td>868736817875</td>
<td>120</td>
<td>868736817834</td>
<td>41</td>
<td>20.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>248</td>
<td>121</td>
<td>868736819998</td>
<td>121</td>
<td>868736819958</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>249</td>
<td>122</td>
<td>868736822101</td>
<td>122</td>
<td>868736822071</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>123</td>
<td>868736824234</td>
<td>123</td>
<td>868736824184</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>251</td>
<td>124</td>
<td>868736826357</td>
<td>124</td>
<td>868736826307</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>252</td>
<td>125</td>
<td>868736828470</td>
<td>125</td>
<td>868736828420</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>253</td>
<td>0</td>
<td>868736830563</td>
<td>0</td>
<td>868736830533</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>254</td>
<td>1</td>
<td>868736832746</td>
<td>1</td>
<td>868736832656</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>255</td>
<td>2</td>
<td>868736834869</td>
<td>2</td>
<td>868736834769</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>3</td>
<td>868736836992</td>
<td>3</td>
<td>868736836892</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>257</td>
<td>4</td>
<td>868736839105</td>
<td>4</td>
<td>868736839005</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>258</td>
<td>5</td>
<td>868736841208</td>
<td>5</td>
<td>868736841128</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>259</td>
<td>6</td>
<td>868736843331</td>
<td>6</td>
<td>868736843241</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>260</td>
<td>7</td>
<td>868736845394</td>
<td>7</td>
<td>868736845354</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>261</td>
<td>8</td>
<td>868736847497</td>
<td>8</td>
<td>868736847477</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>262</td>
<td>9</td>
<td>868736849630</td>
<td>9</td>
<td>868736849590</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>263</td>
<td>10</td>
<td>868736851733</td>
<td>10</td>
<td>868736851713</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>264</td>
<td>11</td>
<td>868736853846</td>
<td>11</td>
<td>868736853826</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>265</td>
<td>12</td>
<td>868736855969</td>
<td>12</td>
<td>868736855939</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>13</td>
<td>868736858172</td>
<td>13</td>
<td>868736858072</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>267</td>
<td>14</td>
<td>868736860296</td>
<td>14</td>
<td>868736860185</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>268</td>
<td>15</td>
<td>868736862409</td>
<td>15</td>
<td>868736862298</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>269</td>
<td>16</td>
<td>868736864512</td>
<td>16</td>
<td>868736864421</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>270</td>
<td>17</td>
<td>868736866615</td>
<td>17</td>
<td>868736866534</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>271</td>
<td>18</td>
<td>868736868738</td>
<td>18</td>
<td>868736868658</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>272</td>
<td>19</td>
<td>868736870821</td>
<td>19</td>
<td>868736870771</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>273</td>
<td>20</td>
<td>868736872954</td>
<td>20</td>
<td>868736872884</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>274</td>
<td>21</td>
<td>868736875057</td>
<td>21</td>
<td>868736875007</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>275</td>
<td>22</td>
<td>868736877170</td>
<td>22</td>
<td>868736877120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>276</td>
<td>23</td>
<td>868736879283</td>
<td>23</td>
<td>868736879233</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>277</td>
<td>24</td>
<td>868736881446</td>
<td>24</td>
<td>868736881356</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>278</td>
<td>25</td>
<td>868736883549</td>
<td>25</td>
<td>868736883469</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>279</td>
<td>26</td>
<td>868736885672</td>
<td>26</td>
<td>868736885582</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>280</td>
<td>27</td>
<td>868736887775</td>
<td>27</td>
<td>868736887705</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>281</td>
<td>28</td>
<td>868736889998</td>
<td>28</td>
<td>868736889818</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>282</td>
<td>29</td>
<td>868736892001</td>
<td>29</td>
<td>868736891941</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>283</td>
<td>30</td>
<td>868736894114</td>
<td>30</td>
<td>868736894054</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>284</td>
<td>31</td>
<td>868736896227</td>
<td>31</td>
<td>868736896167</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>285</td>
<td>32</td>
<td>868736898340</td>
<td>32</td>
<td>868736898290</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>286</td>
<td>33</td>
<td>868736900443</td>
<td>33</td>
<td>868736900403</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>287</td>
<td>34</td>
<td>868736902556</td>
<td>34</td>
<td>868736902516</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>288</td>
<td>35</td>
<td>868736904679</td>
<td>35</td>
<td>868736904639</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>289</td>
<td>36</td>
<td>868736906822</td>
<td>36</td>
<td>868736906752</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>290</td>
<td>37</td>
<td>868736908935</td>
<td>37</td>
<td>868736908865</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>291</td>
<td>38</td>
<td>868736911048</td>
<td>38</td>
<td>868736910988</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>292</td>
<td>39</td>
<td>868736913182</td>
<td>39</td>
<td>868736913101</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>293</td>
<td>40</td>
<td>868736915265</td>
<td>40</td>
<td>868736915214</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>294</td>
<td>41</td>
<td>868736917378</td>
<td>41</td>
<td>868736917338</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>295</td>
<td>42</td>
<td>868736919491</td>
<td>42</td>
<td>868736919451</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>296</td>
<td>43</td>
<td>868736921614</td>
<td>43</td>
<td>868736921574</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>297</td>
<td>44</td>
<td>868736923717</td>
<td>44</td>
<td>868736923687</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>298</td>
<td>45</td>
<td>868736925850</td>
<td>45</td>
<td>868736925810</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>299</td>
<td>46</td>
<td>868736927963</td>
<td>46</td>
<td>868736927923</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>47</td>
<td>868736930066</td>
<td>47</td>
<td>868736930046</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>301</td>
<td>48</td>
<td>868736932179</td>
<td>48</td>
<td>868736932159</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>302</td>
<td>49</td>
<td>868736934292</td>
<td>49</td>
<td>868736934272</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>303</td>
<td>50</td>
<td>868736936349</td>
<td>50</td>
<td>868736936368</td>
<td>110</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>304</td>
<td>51</td>
<td>868736938608</td>
<td>51</td>
<td>868736938508</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>305</td>
<td>52</td>
<td>868736940721</td>
<td>52</td>
<td>868736940621</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>306</td>
<td>53</td>
<td>868736942824</td>
<td>53</td>
<td>868736942744</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>307</td>
<td>54</td>
<td>868736944937</td>
<td>54</td>
<td>868736944857</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>308</td>
<td>55</td>
<td>868736947050</td>
<td>55</td>
<td>868736946970</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>309</td>
<td>56</td>
<td>868736949163</td>
<td>56</td>
<td>868736949093</td>
<td>70</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>310</td>
<td>57</td>
<td>868736951266</td>
<td>57</td>
<td>868736951206</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>311</td>
<td>58</td>
<td>868736953379</td>
<td>58</td>
<td>868736953329</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>312</td>
<td>59</td>
<td>868736955482</td>
<td>59</td>
<td>868736955442</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>313</td>
<td>60</td>
<td>868736957595</td>
<td>60</td>
<td>868736957565</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>314</td>
<td>61</td>
<td>868736959708</td>
<td>61</td>
<td>868736959678</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>315</td>
<td>62</td>
<td>868736961811</td>
<td>62</td>
<td>868736961791</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>316</td>
<td>63</td>
<td>868736964015</td>
<td>63</td>
<td>868736963915</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>317</td>
<td>64</td>
<td>868736966128</td>
<td>64</td>
<td>868736966028</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>318</td>
<td>65</td>
<td>868736968251</td>
<td>65</td>
<td>868736968151</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>319</td>
<td>66</td>
<td>868736970364</td>
<td>66</td>
<td>868736970264</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>320</td>
<td>67</td>
<td>868736972437</td>
<td>67</td>
<td>868736972387</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>321</td>
<td>68</td>
<td>868736974550</td>
<td>68</td>
<td>868736974500</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>322</td>
<td>69</td>
<td>868736976663</td>
<td>69</td>
<td>868736976613</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>323</td>
<td>70</td>
<td>868736978766</td>
<td>70</td>
<td>868736978736</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>324</td>
<td>71</td>
<td>868736980879</td>
<td>71</td>
<td>868736980849</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>325</td>
<td>72</td>
<td>868736983042</td>
<td>72</td>
<td>868736982962</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>326</td>
<td>73</td>
<td>868736985205</td>
<td>73</td>
<td>868736985085</td>
<td>120</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>327</td>
<td>74</td>
<td>868736987238</td>
<td>74</td>
<td>868736987198</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>328</td>
<td>75</td>
<td>868736989351</td>
<td>75</td>
<td>868736989321</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>329</td>
<td>76</td>
<td>868736991454</td>
<td>76</td>
<td>868736991434</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>330</td>
<td>77</td>
<td>868736993577</td>
<td>77</td>
<td>868736993547</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>331</td>
<td>78</td>
<td>868736995720</td>
<td>78</td>
<td>868736995670</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>332</td>
<td>79</td>
<td>868736997843</td>
<td>79</td>
<td>868736997783</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>333</td>
<td>80</td>
<td>868736999946</td>
<td>80</td>
<td>868736999896</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>334</td>
<td>81</td>
<td>868737002049</td>
<td>81</td>
<td>868737002019</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>335</td>
<td>82</td>
<td>868737004172</td>
<td>82</td>
<td>868737004132</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>336</td>
<td>83</td>
<td>868737006285</td>
<td>83</td>
<td>868737006245</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>337</td>
<td>84</td>
<td>868737008388</td>
<td>84</td>
<td>868737008368</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>338</td>
<td>85</td>
<td>868737010512</td>
<td>85</td>
<td>868737010481</td>
<td>31</td>
<td>15.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>339</td>
<td>86</td>
<td>868737012625</td>
<td>86</td>
<td>868737012605</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>340</td>
<td>87</td>
<td>868737014828</td>
<td>87</td>
<td>868737014718</td>
<td>110</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>341</td>
<td>88</td>
<td>868737016931</td>
<td>88</td>
<td>868737016831</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>342</td>
<td>89</td>
<td>868737018934</td>
<td>89</td>
<td>868737018954</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>343</td>
<td>90</td>
<td>868737021147</td>
<td>90</td>
<td>868737021067</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>344</td>
<td>91</td>
<td>868737023260</td>
<td>91</td>
<td>868737023190</td>
<td>70</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>345</td>
<td>92</td>
<td>868737025363</td>
<td>92</td>
<td>868737025303</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>346</td>
<td>93</td>
<td>868737027466</td>
<td>93</td>
<td>868737027416</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>347</td>
<td>94</td>
<td>868737029579</td>
<td>94</td>
<td>868737029539</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>348</td>
<td>95</td>
<td>868737031682</td>
<td>95</td>
<td>868737031652</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>349</td>
<td>96</td>
<td>868737033785</td>
<td>96</td>
<td>868737033775</td>
<td>10</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>350</td>
<td>97</td>
<td>868737035888</td>
<td>97</td>
<td>868737035888</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>198737038031</td>
<td></td>
<td>198737038001</td>
<td></td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>351</td>
<td>98</td>
<td>198737040134</td>
<td>99</td>
<td>198737040124</td>
<td>10</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>352</td>
<td>99</td>
<td>198737042337</td>
<td>100</td>
<td>198737042237</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>353</td>
<td>100</td>
<td>198737044440</td>
<td>101</td>
<td>198737044360</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>354</td>
<td>101</td>
<td>198737046573</td>
<td>102</td>
<td>198737046473</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>355</td>
<td>102</td>
<td>198737048676</td>
<td>103</td>
<td>198737048596</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>356</td>
<td>103</td>
<td>198737050779</td>
<td>104</td>
<td>198737050719</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>357</td>
<td>104</td>
<td>198737052892</td>
<td>105</td>
<td>198737052832</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>358</td>
<td>105</td>
<td>198737055005</td>
<td>106</td>
<td>198737054945</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>359</td>
<td>106</td>
<td>198737057100</td>
<td>107</td>
<td>198737057068</td>
<td>41</td>
<td>20.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>360</td>
<td>107</td>
<td>198737059222</td>
<td>108</td>
<td>198737059181</td>
<td>41</td>
<td>20.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>361</td>
<td>108</td>
<td>198737061385</td>
<td>109</td>
<td>198737061295</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>362</td>
<td>109</td>
<td>198737063508</td>
<td>110</td>
<td>198737063418</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>363</td>
<td>110</td>
<td>198737065631</td>
<td>111</td>
<td>198737065531</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>364</td>
<td>111</td>
<td>198737067754</td>
<td>112</td>
<td>198737067654</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>365</td>
<td>112</td>
<td>198737069857</td>
<td>113</td>
<td>198737069767</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>366</td>
<td>113</td>
<td>198737071970</td>
<td>114</td>
<td>198737071880</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>367</td>
<td>114</td>
<td>198737074083</td>
<td>115</td>
<td>198737074003</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>368</td>
<td>115</td>
<td>198737076119</td>
<td>116</td>
<td>198737076116</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>369</td>
<td>116</td>
<td>198737078229</td>
<td>117</td>
<td>198737078239</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>370</td>
<td>117</td>
<td>198737080412</td>
<td>118</td>
<td>198737080352</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>371</td>
<td>118</td>
<td>198737082525</td>
<td>119</td>
<td>198737082465</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>372</td>
<td>119</td>
<td>198737084628</td>
<td>120</td>
<td>198737084578</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>373</td>
<td>120</td>
<td>198737086731</td>
<td>121</td>
<td>198737086701</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>374</td>
<td>121</td>
<td>198737088844</td>
<td>122</td>
<td>198737088814</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>375</td>
<td>122</td>
<td>198737090957</td>
<td>123</td>
<td>198737090937</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>376</td>
<td>123</td>
<td>198737093080</td>
<td>124</td>
<td>198737093050</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>377</td>
<td>124</td>
<td>198737095193</td>
<td>125</td>
<td>198737095173</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>378</td>
<td>125</td>
<td>198737097326</td>
<td>0</td>
<td>198737097286</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>379</td>
<td>0</td>
<td>198737099449</td>
<td>1</td>
<td>198737099399</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>380</td>
<td>1</td>
<td>198737101582</td>
<td>2</td>
<td>198737101522</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>381</td>
<td>2</td>
<td>198737103685</td>
<td>3</td>
<td>198737103635</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>382</td>
<td>3</td>
<td>198737105809</td>
<td>4</td>
<td>198737105748</td>
<td>61</td>
<td>30.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>383</td>
<td>4</td>
<td>198737107922</td>
<td>5</td>
<td>198737107872</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>384</td>
<td>5</td>
<td>198737110085</td>
<td>6</td>
<td>198737110985</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>385</td>
<td>6</td>
<td>198737112188</td>
<td>7</td>
<td>198737112098</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>386</td>
<td>7</td>
<td>198737114311</td>
<td>8</td>
<td>198737114221</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>387</td>
<td>8</td>
<td>198737116364</td>
<td>9</td>
<td>198737116334</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>388</td>
<td>9</td>
<td>198737118487</td>
<td>10</td>
<td>198737118457</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>389</td>
<td>10</td>
<td>198737120600</td>
<td>11</td>
<td>198737120570</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>390</td>
<td>11</td>
<td>198737122703</td>
<td>12</td>
<td>198737122683</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>391</td>
<td>12</td>
<td>198737124826</td>
<td>13</td>
<td>198737124806</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>392</td>
<td>13</td>
<td>198737126949</td>
<td>14</td>
<td>198737126919</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>393</td>
<td>14</td>
<td>198737129062</td>
<td>15</td>
<td>198737129042</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>394</td>
<td>15</td>
<td>198737131265</td>
<td>16</td>
<td>198737131155</td>
<td>110</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>395</td>
<td>16</td>
<td>198737133368</td>
<td>17</td>
<td>198737133268</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>396</td>
<td>17</td>
<td>198737135481</td>
<td>18</td>
<td>198737135391</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>397</td>
<td>18</td>
<td>198737137614</td>
<td>19</td>
<td>198737137504</td>
<td>110</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>398</td>
<td>19</td>
<td>198737139717</td>
<td>20</td>
<td>198737139617</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>399</td>
<td>20</td>
<td>198737141830</td>
<td>21</td>
<td>198737141740</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>21</td>
<td>198737143953</td>
<td>22</td>
<td>198737143853</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>402</td>
<td>23</td>
<td>868737146016</td>
<td>23</td>
<td>868737145976</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>--------------</td>
<td>----</td>
<td>--------------</td>
<td>----</td>
<td>-----</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>403</td>
<td>24</td>
<td>868737148139</td>
<td>24</td>
<td>868737148089</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>404</td>
<td>25</td>
<td>868737150262</td>
<td>25</td>
<td>868737150202</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>405</td>
<td>26</td>
<td>868737152365</td>
<td>26</td>
<td>868737152325</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>406</td>
<td>27</td>
<td>868737154479</td>
<td>27</td>
<td>868737154438</td>
<td>41</td>
<td>20.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>407</td>
<td>28</td>
<td>868737156582</td>
<td>28</td>
<td>868737156552</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>408</td>
<td>29</td>
<td>868737158695</td>
<td>29</td>
<td>868737158665</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>409</td>
<td>30</td>
<td>868737160808</td>
<td>30</td>
<td>868737160788</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>410</td>
<td>31</td>
<td>868737162981</td>
<td>31</td>
<td>868737162901</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>411</td>
<td>32</td>
<td>868737165094</td>
<td>32</td>
<td>868737165024</td>
<td>70</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>412</td>
<td>33</td>
<td>868737167227</td>
<td>33</td>
<td>868737167137</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>413</td>
<td>34</td>
<td>868737169330</td>
<td>34</td>
<td>868737169250</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>414</td>
<td>35</td>
<td>868737171433</td>
<td>35</td>
<td>868737171373</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>415</td>
<td>36</td>
<td>868737173556</td>
<td>36</td>
<td>868737173486</td>
<td>70</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>416</td>
<td>37</td>
<td>868737175639</td>
<td>37</td>
<td>868737175609</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>417</td>
<td>38</td>
<td>868737177752</td>
<td>38</td>
<td>868737177722</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>418</td>
<td>39</td>
<td>868737179865</td>
<td>39</td>
<td>868737179835</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>419</td>
<td>40</td>
<td>868737181978</td>
<td>40</td>
<td>868737181958</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>420</td>
<td>41</td>
<td>868737184091</td>
<td>41</td>
<td>868737184081</td>
<td>10</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>421</td>
<td>42</td>
<td>868737186214</td>
<td>42</td>
<td>868737186194</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>422</td>
<td>43</td>
<td>868737188337</td>
<td>43</td>
<td>868737188307</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>423</td>
<td>44</td>
<td>868737190460</td>
<td>44</td>
<td>868737190430</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>424</td>
<td>45</td>
<td>868737192573</td>
<td>45</td>
<td>868737192543</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>425</td>
<td>46</td>
<td>868737194696</td>
<td>46</td>
<td>868737194666</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>426</td>
<td>47</td>
<td>868737196839</td>
<td>47</td>
<td>868737196779</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>427</td>
<td>48</td>
<td>868737199003</td>
<td>48</td>
<td>868737198992</td>
<td>111</td>
<td>55.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>428</td>
<td>49</td>
<td>868737201035</td>
<td>49</td>
<td>868737201015</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>429</td>
<td>50</td>
<td>868737203179</td>
<td>50</td>
<td>868737203128</td>
<td>51</td>
<td>25.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>430</td>
<td>51</td>
<td>868737205372</td>
<td>51</td>
<td>868737205252</td>
<td>120</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>431</td>
<td>52</td>
<td>868737207425</td>
<td>52</td>
<td>868737207365</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>432</td>
<td>53</td>
<td>868737209608</td>
<td>53</td>
<td>868737209488</td>
<td>120</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>433</td>
<td>54</td>
<td>868737211621</td>
<td>54</td>
<td>868737211601</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>434</td>
<td>55</td>
<td>868737213864</td>
<td>55</td>
<td>868737213714</td>
<td>150</td>
<td>75</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>435</td>
<td>56</td>
<td>868737215897</td>
<td>56</td>
<td>868737215837</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>436</td>
<td>57</td>
<td>868737218010</td>
<td>57</td>
<td>868737217950</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>437</td>
<td>58</td>
<td>868737220203</td>
<td>58</td>
<td>868737220063</td>
<td>140</td>
<td>70</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>438</td>
<td>59</td>
<td>868737222216</td>
<td>59</td>
<td>868737222186</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>439</td>
<td>60</td>
<td>868737224409</td>
<td>60</td>
<td>868737224299</td>
<td>110</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>440</td>
<td>61</td>
<td>868737226462</td>
<td>61</td>
<td>868737226422</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>441</td>
<td>62</td>
<td>868737228665</td>
<td>62</td>
<td>868737228535</td>
<td>130</td>
<td>65</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>442</td>
<td>63</td>
<td>868737230758</td>
<td>63</td>
<td>868737230648</td>
<td>110</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>443</td>
<td>64</td>
<td>868737232881</td>
<td>64</td>
<td>868737232771</td>
<td>110</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>444</td>
<td>65</td>
<td>868737234944</td>
<td>65</td>
<td>868737234884</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>445</td>
<td>66</td>
<td>868737237107</td>
<td>66</td>
<td>868737236997</td>
<td>110</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>446</td>
<td>67</td>
<td>868737239240</td>
<td>67</td>
<td>868737239120</td>
<td>120</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>447</td>
<td>68</td>
<td>868737241273</td>
<td>68</td>
<td>868737241233</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>448</td>
<td>69</td>
<td>868737243376</td>
<td>69</td>
<td>868737243346</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>449</td>
<td>70</td>
<td>868737245539</td>
<td>70</td>
<td>868737245469</td>
<td>70</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>450</td>
<td>71</td>
<td>868737247703</td>
<td>71</td>
<td>868737247582</td>
<td>121</td>
<td>60.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>451</td>
<td>72</td>
<td>868737249826</td>
<td>72</td>
<td>868737249695</td>
<td>131</td>
<td>65.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>452</td>
<td>73</td>
<td>868737251849</td>
<td>73</td>
<td>868737251818</td>
<td>31</td>
<td>15.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>453</td>
<td>74</td>
<td>868737253982</td>
<td>74</td>
<td>868737253932</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>454</td>
<td>75</td>
<td>868737256155</td>
<td>75</td>
<td>868737256045</td>
<td>110</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>455</td>
<td>76</td>
<td>868737258238</td>
<td>76</td>
<td>868737258168</td>
<td>70</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>456</td>
<td>77</td>
<td>868737260431</td>
<td>77</td>
<td>868737260281</td>
<td>150</td>
<td>75</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>457</td>
<td>78</td>
<td>868737262444</td>
<td>78</td>
<td>868737262404</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>458</td>
<td>79</td>
<td>868737264687</td>
<td>79</td>
<td>868737264517</td>
<td>170</td>
<td>85</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>459</td>
<td>80</td>
<td>868737266700</td>
<td>80</td>
<td>868737266640</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>460</td>
<td>81</td>
<td>868737268833</td>
<td>81</td>
<td>868737268753</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>461</td>
<td>82</td>
<td>868737270886</td>
<td>82</td>
<td>868737270866</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>462</td>
<td>83</td>
<td>868737273079</td>
<td>83</td>
<td>868737272979</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>463</td>
<td>84</td>
<td>868737275202</td>
<td>84</td>
<td>868737275102</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>464</td>
<td>85</td>
<td>8687372777235</td>
<td>85</td>
<td>8687372777215</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>465</td>
<td>86</td>
<td>868737279368</td>
<td>86</td>
<td>868737279338</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>466</td>
<td>87</td>
<td>868737281155</td>
<td>87</td>
<td>868737281451</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>467</td>
<td>88</td>
<td>868737283604</td>
<td>88</td>
<td>868737283564</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>468</td>
<td>89</td>
<td>868737285747</td>
<td>89</td>
<td>868737285687</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>469</td>
<td>90</td>
<td>868737287910</td>
<td>90</td>
<td>868737287800</td>
<td>110</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>470</td>
<td>91</td>
<td>868737289973</td>
<td>91</td>
<td>868737289913</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>471</td>
<td>92</td>
<td>868737292146</td>
<td>92</td>
<td>868737292036</td>
<td>110</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>472</td>
<td>93</td>
<td>868737294209</td>
<td>93</td>
<td>868737294149</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>473</td>
<td>94</td>
<td>868737296292</td>
<td>94</td>
<td>868737296272</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>474</td>
<td>95</td>
<td>868737298486</td>
<td>95</td>
<td>868737298385</td>
<td>101</td>
<td>50.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>475</td>
<td>96</td>
<td>868737300699</td>
<td>96</td>
<td>868737300498</td>
<td>201</td>
<td>100.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>476</td>
<td>97</td>
<td>868737302702</td>
<td>97</td>
<td>868737302622</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>477</td>
<td>98</td>
<td>868737304815</td>
<td>98</td>
<td>868737304735</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>478</td>
<td>99</td>
<td>868737306918</td>
<td>99</td>
<td>868737306848</td>
<td>70</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>479</td>
<td>100</td>
<td>868737309061</td>
<td>100</td>
<td>868737308971</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>480</td>
<td>101</td>
<td>868737311134</td>
<td>101</td>
<td>868737311084</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>481</td>
<td>102</td>
<td>868737313297</td>
<td>102</td>
<td>868737313197</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>482</td>
<td>103</td>
<td>868737315390</td>
<td>103</td>
<td>868737315320</td>
<td>70</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>483</td>
<td>104</td>
<td>868737317553</td>
<td>104</td>
<td>868737317433</td>
<td>120</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>484</td>
<td>105</td>
<td>868737319596</td>
<td>105</td>
<td>868737319546</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>485</td>
<td>106</td>
<td>868737321769</td>
<td>106</td>
<td>868737321669</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>486</td>
<td>107</td>
<td>868737323842</td>
<td>107</td>
<td>868737323782</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>487</td>
<td>108</td>
<td>868737325895</td>
<td>108</td>
<td>868737325895</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>488</td>
<td>109</td>
<td>868737328128</td>
<td>109</td>
<td>868737328018</td>
<td>110</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>489</td>
<td>110</td>
<td>868737330271</td>
<td>110</td>
<td>868737330131</td>
<td>140</td>
<td>70</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>490</td>
<td>111</td>
<td>868737332344</td>
<td>111</td>
<td>868737332244</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>491</td>
<td>112</td>
<td>868737334397</td>
<td>112</td>
<td>868737334367</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492</td>
<td>113</td>
<td>868737336590</td>
<td>113</td>
<td>868737336480</td>
<td>110</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>493</td>
<td>114</td>
<td>868737338723</td>
<td>114</td>
<td>868737338593</td>
<td>130</td>
<td>65</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>494</td>
<td>115</td>
<td>868737340836</td>
<td>115</td>
<td>868737340716</td>
<td>120</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>495</td>
<td>116</td>
<td>868737342850</td>
<td>116</td>
<td>868737342829</td>
<td>121</td>
<td>60.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>496</td>
<td>117</td>
<td>868737345012</td>
<td>117</td>
<td>868737344942</td>
<td>70</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>497</td>
<td>118</td>
<td>868737347116</td>
<td>118</td>
<td>868737347065</td>
<td>51</td>
<td>25.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>498</td>
<td>119</td>
<td>868737349299</td>
<td>119</td>
<td>868737349178</td>
<td>121</td>
<td>60.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>499</td>
<td>120</td>
<td>868737351422</td>
<td>120</td>
<td>868737351292</td>
<td>130</td>
<td>65</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>121</td>
<td>868737353485</td>
<td>121</td>
<td>868737353415</td>
<td>70</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>501</td>
<td>122</td>
<td>868737355548</td>
<td>122</td>
<td>868737355528</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>502</td>
<td>123</td>
<td>868737357761</td>
<td>123</td>
<td>868737357641</td>
<td>120</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>503</td>
<td>124</td>
<td>868737359844</td>
<td>124</td>
<td>868737359764</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>-----</td>
<td>-----------------------</td>
<td>-----</td>
<td>-----------------------</td>
<td>----</td>
<td>------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>504</td>
<td>125</td>
<td>868737361907</td>
<td>125</td>
<td>868737361877</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>505</td>
<td>0</td>
<td>868737364030</td>
<td>0</td>
<td>868737364000</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>506</td>
<td>1</td>
<td>868737366213</td>
<td>1</td>
<td>868737366113</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>507</td>
<td>2</td>
<td>868737368246</td>
<td>2</td>
<td>868737368226</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>508</td>
<td>3</td>
<td>868737370449</td>
<td>3</td>
<td>868737370349</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>509</td>
<td>4</td>
<td>868737372532</td>
<td>4</td>
<td>868737372462</td>
<td>70</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>510</td>
<td>5</td>
<td>868737374685</td>
<td>5</td>
<td>868737374685</td>
<td>100</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>511</td>
<td>6</td>
<td>868737376818</td>
<td>6</td>
<td>868737376698</td>
<td>120</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>7</td>
<td>868737378931</td>
<td>7</td>
<td>868737378811</td>
<td>120</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>513</td>
<td>8</td>
<td>868737380984</td>
<td>8</td>
<td>868737380934</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>514</td>
<td>9</td>
<td>868737383157</td>
<td>9</td>
<td>868737383047</td>
<td>110</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>515</td>
<td>10</td>
<td>868737385190</td>
<td>10</td>
<td>868737385160</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>516</td>
<td>11</td>
<td>868737387303</td>
<td>11</td>
<td>868737387283</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>517</td>
<td>12</td>
<td>868737389476</td>
<td>12</td>
<td>868737389396</td>
<td>80</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>518</td>
<td>13</td>
<td>868737391640</td>
<td>13</td>
<td>868737391519</td>
<td>121</td>
<td>60.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>519</td>
<td>14</td>
<td>868737393672</td>
<td>14</td>
<td>868737393632</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>520</td>
<td>15</td>
<td>868737395786</td>
<td>15</td>
<td>868737395745</td>
<td>41</td>
<td>20.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>521</td>
<td>16</td>
<td>868737397959</td>
<td>16</td>
<td>868737397868</td>
<td>91</td>
<td>45.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>522</td>
<td>17</td>
<td>868737400102</td>
<td>17</td>
<td>868737399982</td>
<td>120</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>523</td>
<td>18</td>
<td>868737402245</td>
<td>18</td>
<td>868737402095</td>
<td>150</td>
<td>75</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>524</td>
<td>19</td>
<td>868737404278</td>
<td>19</td>
<td>868737404218</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>525</td>
<td>20</td>
<td>868737406351</td>
<td>20</td>
<td>868737406331</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>526</td>
<td>21</td>
<td>868737408504</td>
<td>21</td>
<td>868737408444</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>527</td>
<td>22</td>
<td>868737410687</td>
<td>22</td>
<td>868737410567</td>
<td>120</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>528</td>
<td>23</td>
<td>868737412700</td>
<td>23</td>
<td>868737412680</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>529</td>
<td>24</td>
<td>868737414873</td>
<td>24</td>
<td>868737414803</td>
<td>70</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>530</td>
<td>25</td>
<td>868737417046</td>
<td>25</td>
<td>868737416916</td>
<td>130</td>
<td>65</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>531</td>
<td>26</td>
<td>868737419119</td>
<td>26</td>
<td>868737419029</td>
<td>90</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>532</td>
<td>27</td>
<td>868737421272</td>
<td>27</td>
<td>868737421142</td>
<td>130</td>
<td>65</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>533</td>
<td>28</td>
<td>868737423315</td>
<td>28</td>
<td>868737423265</td>
<td>50</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>534</td>
<td>29</td>
<td>868737425498</td>
<td>29</td>
<td>868737425378</td>
<td>120</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>535</td>
<td>30</td>
<td>868737427561</td>
<td>30</td>
<td>868737427501</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center .................................................. 2
   8725 John J. Kingman Road, Suite 0944
   Fort Belvoir, VA 22060-6218

2. Dudley Knox Library ........................................................................ 2
   Naval Postgraduate School
   Monterey, CA 93943

3. Center for Naval Analysis .................................................................. 1
   4401 Ford Ave.
   Alexandria, VA 22302

4. Dr. Ted Lewis, Chairman, Code CS/L ............................................. 1
   Computer Science Dept.
   Naval Postgraduate School
   Monterey, CA 93943

5. Chief of Naval Research .................................................................. 1
   800 North Quincy St.
   Arlington, VA 22217

6. Dr. Luqi, Code CS/Lq ................................................................. 1
   Computer Science Dept.
   Naval Postgraduate School
   Monterey, CA 93943

7. Dr. Marvin Langston ................................................................. 1
   1225 Jefferson Davis Highway
   Crystal Gateway 2 / Suite 1500
   Arlington, VA 22202-4311

8. David Hislop ............................................................................... 1
   U.S. Army Research Office
   PO Box 12211
   Research Triangle Park, NC 27709-2211

9. Capt. Talbot Manvel ................................................................. 1
   Naval Sea System Command
   2531 Jefferson Davis Hwy.
   Attn: TMS 378 Capt. Manvel
   Arlington, VA 22240-5150
10. CDR Michael McMahon
    Naval Sea System Command
    2531 Jefferson Davis Hwy.
    Arlington, VA 22242-5160

11. Dr. Elizabeth Wald
    Office Of Naval Research
    800 N. Quincy St.
    ONR CODE 311
    Arlington, VA 22217-5660

12. Dr. Ralph Wachter
    Office of Naval Research
    800 N. Quincy St.
    CODE 311
    Arlington, VA 22217-5660

13. Army Research Lab
    115 O'Keefe Building
    Attn: Mark Kendall
    Atlanta, GA 30332-0862

14. National Science Foundation
    Attn: Bruce Barnes
    Div. Computer & Computation Research
    1800 G St. NW
    Washington, DC 20550

15. National Science Foundation
    Attn: Bill Agresty
    4201 Wilson Blvd.
    Arlington, VA 22230

    Assistant Secretary of the Navy
    (Research, Development and Acquisition)
    Room E741
    1000 Navy Pentagon
    Washington, DC 20350-1000

17. VADM Herbert A. Brown, Commander Third Fleet
    2421 Vella La Vella Rd.
    NAB Coronado
    San Diego, CA 92155-5490
18. RADM George F. A. Wagner, Code 00 .................................................. 1
    SPAWAR
    53560 Oceanview Dr. Ste.317
    San Diego, CA 92152-5088

19. Paul Wessel, Code 07 ................................................................. 1
    SPAWAR
    53560 Oceanview Dr. Ste.317
    San Diego, CA 92152-5088

20. Capt. Harold Williams, Code D00 ............................................. 1
    NCCOSC RDT&E Division
    53560 Hull Street
    San Diego, CA 92152-5001

21. Dr. Robert Kolb, Code D01 ...................................................... 1
    NCCOSC RDT&E Division
    53560 Hull Street
    San Diego, CA 92152-5001

22. Rod Smith, Code D40 ............................................................. 1
    NCCOSC RDT&E Division
    53560 Hull Street
    San Diego, CA 92152-5001

23. Jim Price, Code D43 ............................................................. 1
    NCCOSC RDT&E Division
    53560 Hull Street
    San Diego, CA 92152-5001

24. Rick Millen, Code D431 .......................................................... 1
    NCCOSC RDT&E Division
    53560 Hull Street
    San Diego, CA 92152-5001

25. Rich Heidecker, Code D432 ................................................... 1
    NCCOSC RDT&E Division
    53560 Hull Street
    San Diego, CA 92152-5001

26. Floyd Bailey, Code D4322 ...................................................... 4
    NCCOSC RDT&E Division
    53560 Hull Street
    San Diego, CA 92152-5001
27. Carl Robbins, Code D4323 ................................................................. 4
    NCCOSC RDT&E Division
    53560 Hull Street
    San Diego, CA 92152-5001

28. Technical Library Branch, Code D0274 .......................................... 1
    NCCOSC RDT&E Division
    53560 Hull Street
    San Diego, CA 92152-5001