AD-A101 490 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH F/G 9/2
A TOOL FOR DETECTING PLAGIARISM IN PASCAL PROGRAMS.(U)
DEC 80 S L GRIER
UNCLASSIFIED AFIT-CI-80-74T NL

1 OF 2
AD A
101490

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|
| 1. REPORT NUMBER 80-74T AFIT CI-80-74T | 2. GOVT ACCESSION NO. AD-A101490 |
| | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle) A Tool for Detecting Plagiarism in Pascal Programs. | 5. TYPE OF REPORT & PERIOD COVERED THESIS/DISSERTATION |
| | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s) Samuel L. Grier, Jr | 8. CONTRACT OR GRANT NUMBER(s) |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS AFIT STUDENT AT: University of Colorado | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS AFIT/NR WPAFB OH 45433 | 12. REPORT DATE December 1980 |
| | 13. NUMBER OF PAGES 111 |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) | 15. SECURITY CLASS. (of this report) UNCLASS |
| | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

LEVEL II

ADA101490

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC ELECTE JUL 17 1981 S D F

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

APPROVED FOR PUBLIC RELEASE: IAW AFR 190-17

23 JUN 1981

FREDRIC C. LYNCH, Major, USAF
Director of Public Affairs
Air Force Institute of Technology (ATC)
Wright-Patterson AFB, OH 45433

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

ATTACHED

81 7 16 028

DTIC FILE COPY

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

# AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to ascertain the value and/or contribution of research accomplished by students or faculty of the Air Force Institute of Technology (ATC). It would be greatly appreciated if you would complete the following questionnaire and return it to:

AFIT/NR
Wright-Patterson AFB OH 45433

RESEARCH TITLE: A Tool for Detecting Plagiarism in Pascal Programs

AUTHOR: Samuel L. Grier, Jr.

RESEARCH ASSESSMENT QUESTIONS:

1. Did this research contribute to a current Air Force project?

( ) a. YES ( ) b. NO

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not?

( ) a. YES ( ) b. NO

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency achieved/received by virtue of AFIT performing the research. Can you estimate what this research would have cost if it had been accomplished under contract or if it had been done in-house in terms of manpower and/or dollars?

( ) a. MAN-YEARS ________ ( ) b. $________

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3. above), what is your estimate of its significance?

( ) a. HIGHLY SIGNIFICANT ( ) b. SIGNIFICANT ( ) c. SLIGHTLY SIGNIFICANT ( ) d. OF NO SIGNIFICANCE

5. AFIT welcomes any further comments you may have on the above questions, or any additional details concerning the current application, future potential, or other value of this research. Please use the bottom part of this questionnaire for your statement(s).

NAME ________ GRADE ________ POSITION ________

ORGANIZATION ________ LOCATION ________

STATEMENT(s):

FOLD DOWN ON OUTSIDE - SEAL WITH TAPE

AFIT/NR
WRIGHT-PATTERSON AFB OH 45433

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, $300

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 73236 WASHINGTON D.C.

POSTAGE WILL BE PAID BY ADDRESSEE

AFIT/ DAA
Wright-Patterson AFB OH 45433

FOLD IN

A TOOL FOR DETECTING PLAGIARISM IN PASCAL PROGRAMS

by

Samuel L. Grier, Jr.

B.S., USAF Academy, 1973

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Computer Science

1980

This Thesis for the Master of Science Degree by

Samuel L. Grier, Jr.

has been approved for the

Department of

Computer Science

by

Lloyd D. Fosdick

Lloyd D. Fosdick

Malcolm Newey

Malcolm C. Newey

H. Paul Zeiger

H. Paul Zeiger

Date 5 DEC 80

Grier, Samuel L., Jr. (M.S., Computer Science)
A Tool for Detecting Plagiarism in Pascal Programs
Thesis directed by Professor Lloyd D. Fosdick

Plagiarism has become a problem in introductory Computer Science courses. Programmed assignments can be copied and transformed with little human effort. A pertinent recommendation has resulted from this realization: an on-line system to detect programs that are "too similar" and hence suspected of plagiarism should be developed [5]. The purpose of this thesis has been to construct such a system in the form of Program Accuse.

Program Accuse analyzes Pascal programs to detect those pairs of programs such that plagiarism is a possibility.

An overriding concern of the development of Accuse has been that it be inexpensive to use. In addition, the use of Accuse is intended for introductory Computer Science courses. The result is a program that is efficient, but limited in its ability to detect sophisticated plagiarism. Efficiency means low cost; lack of comprehensive analysis is rationalized with the assumption that the student clever enough to plagiarize with sophistication has no need to plagiarize.

Accuse measures 20 parameters in each program: for example, total lines in the program, variables

declared and not used, and the number of control statements. Seven of these parameters were chosen through testing as a means to compute a correlation number that determines if two programs are similar.

If two programs are considered similar, they are flagged for the user to inspect and make the judgement as to whether plagiarism occurred.

Signed Lloyd D. Fosdick

Faculty member in charge of thesis

# TABLE OF CONTENTS


| CHAPTER | | PAGE |
|---|---|---|
| I. | INTRODUCTION | 1 |
| II. | BACKGROUND | 3 |
| III. | DESIGN OF ACCUSE | 6 |
| IV. | SHORTCOMINGS | 12 |
| V. | OUTPUT | 14 |
| VI. | DEFINING THE CORRELATION SCHEME | 15 |
| VII. | ANALYSIS OF RESULTS | 18 |
| | Effectiveness | 18 |
| | When Plagiarism Occurs | 21 |
| | Side Issues | 25 |
| VIII. | CONCLUSION | 27 |
| SELECTED BIBLIOGRAPHY | | 28 |
| APPENDIX A | | 29 |
| APPENDIX B | | 31 |
| APPENDIX C | | 37 |
| APPENDIX D | | 44 |
| APPENDIX E | | 48 |
| APPENDIX F | | 51 |
| APPENDIX G | | 54 |
| APPENDIX H | | 55 |
| APPENDIX I | | 105 |
| APPENDIX J | | 111 |

# CHAPTER I

## INTRODUCTION

Plagiarism has become a problem in introductory Computer Science courses. Programmed assignments can be copied and transformed with little human effort. A pertinent recommendation has resulted from this realization: an on-line system to detect programs that are "too similar" and hence suspected of plagiarism should be developed [5]. The purpose of this thesis has been to construct such a system in the form of Program Accuse.

Program Accuse analyzes Pascal programs to detect those pairs of programs such that plagiarism is a possibility.

An overriding concern of the development of Accuse has been that it be inexpensive to use. In addition, the use of Accuse is intended for introductory Computer Science courses. The result is a program that is efficient, but limited in its ability to detect sophisticated plagiarism. Efficiency means low cost; lack of comprehensive analysis is rationalized with the assumption that the student clever enough to plagiarize with sophistication has no need to plagiarize.

Accuse measures 20 parameters in each program: for example, total lines in the program, variables declared and not used, and the number of control statements. Seven of these parameters were chosen through testing as a means to compute a correlation number that determines if two programs are similar.

If two programs are considered similar, they are flagged for the user to inspect and make the judgement as to whether plagiarism occurred.

# CHAPTER II

## BACKGROUND

An attempt to construct such an on-line system has been made at Purdue University by K.J. Ottenstein [4]. He developed a program that quantifies the sameness of Fortran programs using the four basic Software Science parameters suggested by M. Halstead as useful measures of program length [3]. These parameters are: (1) the number of unique operators, (2) the number of unique operands, (3) the total number of occurrences of operators, and (4) the total number of occurrences of operands. It seems the first suggestion to use these parameters as measures of similarity or dissimilarity (depending on your viewpoint) came from N. Bulut as a by-product of his study of invariant properties of algorithms [1].

M. Halstead developed the notion of Software Science in 1972. He advances the four parameters above as properties of any computer program that are capable of being counted or measured. He defines these parameters and their relationships as follows [3]:

n1 = number of unique operators

n2 = number of unique operands

N1 = total number of operators

N2 = total number of operands

$$\text{vocabulary } n = n1 + n2$$

$$\text{length } N = N1 + N2$$

He also provides data to support the following relationship [3]:

$$N = n1 \log n1 + n2 \log n2$$

Ottenstein's program utilizes only the four basic Software Science parameters, and it counts them in a straightforward manner. He acknowledges his program detects only cosmetic changes: reordering time independent statements, recommenting, reformatting of text, and renaming variables and labels. He believes that plagiarism can be deterred both by the knowledge of the existence of a program like his and its ability to make it reasonably difficult to cheat successfully [4].

Ottenstein uses the length N to categorize his input programs. Those that have identical N1, N2, n1, and n2 counts are then suspected of plagiarism [4].

Inherent in M. Halstead's theories is the assumption that programs are well-written and polished. For example, in almost all cases for which the length indicator (N) was tested, the programs had been prepared for publication [2].

M. Halstead recognized that not all programs would be well-written, and hence derived and defined six classes of impurities as follows [3]:

(1) complementary operations: the successive application of two complementary operators to the same operand

example: R := T * T + T - T

(2) ambiguous operands: the same operand name is used to represent two or more variables within a program

example: R := P + Q; R := R * R

(3) synonymous operands: using two operand names to represent the same variable within a program

example: T1 := P + Q; T2 := P + Q; R := T1 + T2

(4) common subexpressions: the same subexpression occurs more than one time within a program

example: R := (P*Q) + (P*Q)

(5) unwarranted assignment: an expression is assigned to a temporary operand that is used only once

example: T := P + Q; R := T;

(6) unfactored expressions: the same operators and operands repeat in an expression (making the expression difficult to understand)

example: R := P * P + 2 * P * Q + Q * Q

Fitzsimmons and Love conjecture that a compiler can detect all of these impurities [2], and only (6) above cannot be mechanically corrected [3]. Any system that attempts to detect plagiarism can expect to encounter these impurities.

# CHAPTER III

## DESIGN OF ACCUSE

Two principle ideas guided the development of Accuse: (1) that Accuse be as inexpensive to use as possible, and (2) that the individual able enough to plagiarize cleverly has no need to plagiarize.

When construction of Accuse was being planned, the idea of using the front end of a compiler as the driver was considered. There were several reasons for this: (1) the desire to use as much shelf material as possible, and (2) the lack of awareness of Software Science for this particular application.

After the discovery of Ottenstein's attempt and his method, it was felt that a counter could be written that would be faster than even a stripped down compiler. However, because Accuse is not a compiler, it needs to be used in the context of a larger tool that retrieves programs, compiles them and saves their output for graders, and then sends them to a file for processing by Accuse.

The result of not using a compiler is a compromise between speed and comprehensive analysis. Accuse processes over 170 lines per second. However, as noted above, it will not discover changes made by the sophisticated plagiarist. Again, this is rationalized with the assumption that the student intelligent enough to plagiarize with sophistication has no need to plagiarize.

Accuse was designed top-down, but implemented from the bottom up. Each module was developed as needed; for while we knew the main components of the system, it was impossible to predict the support routines. A module's ability to achieve the desired counts was certified before construction of the next module.

Program Accuse was constructed with the belief that additional parameters are available beyond the four basic Software Science parameters, and that heuristics can be employed to achieve more than detection of cosmetic changes. Using these heuristics and seven parameters, Accuse computes a correlation number that is used to determine the similarity of two programs.

Accuse measures 20 parameters. The seven that comprise the correlation number were selected by testing different combinations of them.

Accuse measures the following 20 parameters (for full definitions see Appendix A):

1. total lines
2. code lines
3. code comment lines
4. multiple statement lines
5. constants and types
6. variables declared (and used)

7. variables declared (and not used)
8. procedures and functions
9. var parameters
10. value parameters
11. procedure variables (includes 9 and 10)
12. for statements
13. repeat statements
14. while statements
15. goto statements
16. unique operators
17. unique operands
18. total operators
19. total operands
20. indenting function

The seven parameters that comprise the correlation number are:

1. unique operators
2. unique operands
3. total operators
4. total operands
5. code lines
6. variables declared (and used)
7. total control statements

While being construrted, it was believed that an "indenting function" would play an important role in the detection of plagiarism. Since Computer Science 210 students use cards and do not have access to the sophisticated editing features of a time sharing terminal, it was thought that changes to the style of a copied program would be clumsy at best. This resulted in the rejection of any sophisticated indenting functions and the selection of a simple one. The function currently counts the number of left, right, and unindented lines of code. The indenting function is created as follows:

```
indenting function =

( (left indentations) mod 1000) * 1000000 +
 (right indentations) mod 1000) * 1000 +
  (zero indentations) mod 1000)
```

The results have proved disappointing. If all of the input programs were processed through a "pretty printer," an indenting function might become important. This additional cost is presently considered prohibitive, and it is contrary to the intent of Accuse being inexpensive to use. The unimportance of an identing function necessitated the search for an alternate parameter that would reflect some characteristic of the lines of a program. The result was the idea to count lines of executable code in a program, and the results of this decision are thus far promising.

The decision to introduce the use of heuristics in the way counts are made in Accuse was two-fold: (1) to make plagiarism difficult to achieve, and (2) to make Accuse's repeated use feasible in light of the fact that its use will quickly become common knowledge. The heuristics are simple and straightforward.

"Total operators" does not include assignment operators. In addition, for every assignment operator found, two operands are subtracted from "total operands," and "code lines" is decremented. The purpose of this is to prevent Accuse from being misled by unnecessary initializations and unnecessary assignment statements. This desire roughly correlates to the prevention of M. Halstead's fifth defined impurity, "unwarranted assignments."

"Code lines" ignores blank lines, comment lines, and declarations. It counts only the lines of executable code within a program. This is intended to prevent excess declarations and comments from affecting this parameter's value.

Accuse is also selective about what it calls operators. A "BEGIN END" combination and "()" combination are considered operators in Software Science. Because BEGINs, ENDs, and parentheses can be added to Pascal code where not required, Accuse chooses to ignore them. A semicolon is ignored for essentially the same reason. IF is considered an operator while THEN is not. ELSE is considered an operator because it is not a necessary part of an IF statement.

As Accuse only counts variables, the obvious tactic of changing variable names makes no difference to Accuse. Since Pascal requires declarations, Accuse can keep track of variables declared and subsequently used or not used. Hence, declaring extra variables and then not using them does not affect Accuse's analysis. Constants of enumerated types and tag fields in case clauses of record declarations that contain a declaration are considered variables. Since these constants cannot be read or written, their nonuse is considered notable.

# CHAPTER IV

## SHORTCOMINGS

Accuse has three main drawbacks. The first is that it is unable to detect five of the impurities defined by Halstead. This may in fact not be that critical; for any system to detect and then "undo" any impurities once found would at the least be expensive; in addition, the individual we wish to catch plagiarizing is not likely to introduce these impurities.

The second is that because the input program is not parsed, but is guided by a driver that expects a compilable program, syntactically incorrect programs may be accepted by Accuse. Accuse uses a modified Pascal scanner, specifically the Pascal-J scanner made available to students at the University of Colorado for graduate work. Hence it detects some syntax errors: for example, incorrect literal strings and comments that lack their left part. However, it may very well accept syntactically incorrect programs.

The final drawback is that since the current policy in conjunction with the use of Accuse does not include the user making the students' "graded runs," there is nothing to prevent a student from changing or sabotaging his program before he submits it for processing by Accuse. The cost of rerunning all students' programs is presently considered prohibitive, and checking every student's final listing against an unordered listing of 150 programs is impractical.

The first drawback is not a detriment if grading enforces a policy that does not allow these impurities by exacting a severe penalty for their use.

The second and third are resolved if Accuse is used in the context of a larger tool.

# CHAPTER V

## OUTPUT

Accuse prints four results for the user. The first is a dump of each program's identifier and its values of the 20 parameters measured by Accuse. This dump is sorted on the "indenting function."

The second result is a dump of each program's identifier and its respective values of the seven parameters used to compute the correlation number; each parameter list is sorted smallest to largest. In the output, the column headed FOR STMT actually contains the total number of control statements. This is the result of the implementation of summing parameters.

The third result is a frequency distribution graph that indicates the number of pairs of programs with like correlation numbers. A new addition to the listings is the Tukey estimate for suspicion of plagiarism.

The final result is a list of all pairs of programs with correlation number greater than or equal to 28. Twenty-nine is currently identified as the number that indicates the possibility of plagiarism, with 32 the maximum correlation number possible.

# CHAPTER VI

## DEFINING THE CORRELATION SCHEME

The scheme that computes the correlation number is only a tentative one. The current scheme was developed and tuned by using a group of 43 programs from an introductory course. Code for three of the programs was written together, but finished individually. The "importance" values for the seven correlation parameters were then adjusted until these three programs were brought into the domain of "those programs suspected of plagiarism."

The current correlation scheme involves computing an increment for each pair of affected programs based on the equation

increment = "importance" - (pcounta - pcountb)

where pcounta and pcountb represent parameter counts and (pcounta - pcountb) is less than or equal to some "window" size depending on the particular parameter.

The computation of the correlation number may well be subject to improvement by a more elaborate scheme or by simple changes to the importance factors.

Five runs of Accuse follow the text of this paper. The first run (Appendix B) processed 13 programs, three of which were input twice. Included in this run is a printout of the triangular matrix that contains correlation values of the pairs of programs. This matrix is not printed in a production model of Accuse.

Below we illustrate the computation of the correlation number for a pair of programs in the first run. Before proceeding, it is necessary to note the following "window" sizes and "importance" factors for each of the correlation parameters:

1. total operators
   window size = 5
   importance factor = 6

2. total operands
   window size = 5
   importance factor = 6

3. unique operators
   window size = 3
   importance factor = 5

4. unique operands
   window size = 3
   importance factor = 5

5. code lines
   window size = 3
   importance factor = 5

6. declared variables (and used)
   window size = 2
   importance factor = 3

7. control statements
   window size = 1
   importance factor = 2

The correlation number for the pair of programs T102 and T107 (see Appendix B, p. 32) is computed as follows:

1. T107 - T102 = 8
   Eight is greater than the window size for this parameter, hence these are not "affected" programs.

2. T107 - T102 = 16
   Again, these are not "affected" programs.

3. T107 - T102 = 1
   These programs are now within the window size, and an increment is calculated for this pair of programs:
   increment = 5 - (25 - 24) = 4
   correlation number = 4

4. T102 - T107 = 0
   increment = 5 - (13 - 13) = 5
   correlation number = 9

5. T102 - T107 = 1
   increment = 5 - (64 - 63) = 4
   correlation number = 13

6. T107 - T102 = 0
   increment = 3 - (11 - 11) = 3
   correlation number = 16

7. T102 - T107 = 0
   increment = 2 - (4 - 4) = 2
   correlation number = 18

The second listing (Appendix C) is a production run of Accuse. There were 137 input programs consisting of 13,374 lines of code. Accuse processed the code on a CDC machine at a cost of $12.32. It required:

```
FL TO LOAD 110700      FL TO RUN 77100
89.956 CP SECS         105237B CM USED
```

The maximum number of asterisks printed in the distribution graph is 40; hence the "flat" distribution.

Accuse prints all pairs of programs with correlation number greater than or equal to 28, though 29 is the number that indicates the possibility of plagiarism.

# CHAPTER VII

# ANALYSIS OF RESULTS

## Effectiveness

A question that arises is, "What are the chances that two programs will be declared similar when they have been independently written?" A similar question is, "How many programs can Accuse accept before so many programs are suspected of plagiarism that Accuse's results become unacceptable?"

These questions are not addressed by Ottenstein. He analyzes his findings and concludes that the way he categorizes the input programs results in a somewhat normal distribution, in agreement with our intuition. He makes the observation that if two programs are suspected of being similar (because they have the same N value), the odds that they are similar are greater if the correlation number occurs at one of the extreme values of N. He concludes that any correlation function that one could derive that produces a constant distribution would not be accurate or necessarily desirable because, in general, meaningful measurements of human behavior produce uneven distributions [4].

I see two aspects to these questions. The first addresses the size of the problem being solved. A problem that takes only 12 lines of code to solve will certainly result in a different answer to these questions than if we consider a problem that takes 100 lines of code to solve. The second considers how many parameters are used to compute the correlation number.

One interesting result of the current data available from Accuse has been that the less code a student writes into a given program, the more even the distribution of the parameters appears to be. Note the third listing (Appendix D). In this assignment students were responsible for approximately 14 lines of code; the rest was given to the student. Ignoring the anomolies, we compute the differences between the minimum and maximum values:

| TOTAL OPERS | TOTAL OPNDS | UNIQ OPERS | UNIQ OPNDS | CODE LINES | DECL VARS | CONT STMTS |
|---|---|---|---|---|---|---|
| 19 | 20 | 4 | 8 | 18 | 8 | 2 |

These compressed ranges imply the occurrences of higher correlation numbers since the correlation numbers are computed using the proximity of values. The frequency distribution graph tells us nine pairs of programs have a correlation number of 28 or higher.

If we go back to the second listing (Appendix C) and again, ignoring anomolies, note the differences between the maximum and minimum values:

| TOTAL OPERS | TOTAL OPNDS | UNIQ OPERS | UNIQ OPNDS | CODE LINES | DECL VARS | CONT STMTS |
|---|---|---|---|---|---|---|
| 48 | 55 | 12 | 11 | 49 | 6 | 4 |

These larger ranges imply the occurrences of lower correlation numbers. The frequency distribution graph tells us six pairs of programs have a correlation number of 28 or higher.

These observations appeal to our intuition. The wider the ranges, the lower the correlation numbers, and vice versa.

Another attractive conjecture is that the more input programs, the higher the correlation numbers generated. In our examples above, our expectation is incorrect. The first set of data where nine pairs of programs correlate at 28 or higher inputs 43 programs. The second inputs 137 programs, and only six pairs of programs correlate at 28 or higher.

We make three assertions: (1) that a simple and short program is going to generate more pairs of programs with high correlation numbers than will a more difficult and longer program when both generate the same number of pairs of programs, (2) that the number of programs that Accuse can accept before its results are unacceptable is a function of both the number of input programs and the complexity and length of those programs, and (3) that the more independent correlation parameters, the lower the correlation numbers.

The first two have already been argued. The third can be argued as follows: let us consider the seven correlation parameters as independent events; for each parameter, one can calculate a theoretical probability that two programs will have the same value; multiplying these seven probabilities together will give the theoretical probability that two programs will have the same value for every parameter; removing any of the given parameters will clearly increase this product, hence increasing the likelihood of two programs having a maximum correlation number.

When Plagiarism Occurs

Available data supports the selection of 29 as the number that suggests plagiarism. This choice was made through observation, and is by no means absolute.

The interesting point of analyzing our data is that we can look at it from two different aspects. The first is as above, where we viewed the results in terms of the individual parameters. Bulut makes the statement that the probability of using n1 and n2 exactly N1 and N2 times in two different algorithms is very slim [1]. Both our results and Ottenstein's results verify his assertion.

The second way to view our results comes from the manner in which we categorize or "fingerprint" the input programs. Ottenstein uses N to categorize his input programs, and it is the distribution that N creates that Ottenstein analyzes. We categorize our programs using a correlation number, and if we analyze the distribution created by our correlation numbers, we come to somewhat the same conclusions.

First, the correlation numbers create a somewhat normal distribution, though they appear not to fit any "standard" distributions [7].

Second, by the way we have built our correlation scheme, two programs are declared similar only if the correlation number occurs at an extreme value of the distribution. In Ottenstein's categorization, two programs can be declared similar in the center of his distribution. Hopefully, then, our correlation scheme is better.

Finally, since the distribution created by our correlation scheme is not a uniform one, it is likely to be an accurate measurement of human behavior [4].

Looking at the data from this viewpoint, it would be nice to have a verification of our selection of 29 as a choice for the number that suggests plagiarism. J.W. Tukey suggests a way to analyze distributions that fit no standard distributions [6]. This analysis fits well with our desire.

He suggests taking two "hinges," one each at the midpoints between the outer edges and the median of the distribution (these hinges correspond to the quartiles). He defines one and one half times the difference between the values that occur at these points as a "step."

Finally, any values that occur beyond the value at these hinges plus two steps (called the "outerfences") are considered unreasonable.

For a hypothetical example, then, if the lower hinge occurs at 14 and the upper hinge at 17, our outerfence occurs at

17 + 2 * (1.5*(17-14)) = 26

and any correlation number greater than 26 is considered unreasonable; or, in our application, considered plagiarism.

Accuse has been altered to compute this value; test results, though inconclusive, are encouraging. Though the fourth listing (Appendix E) provided gives a number of 27 as being the outerfence (hence 28 implies plagiarism), it is easy to see that there are no programs that are beyond the outerfence. One can conclude that in this case, 29 is as good a guess as the computed 28.

Computing the probability that two programs would have the same value for a given parameter was discussed earlier. This computation could lead to supplying the user with some additional information that will help him in his judgement as to whether or not plagiarism has occurred. If we look at the fourth listing, we can make some observations.

First, let us make the assumption that for each range of values for a given parameter, each value has an equal likelihood of occurring. Second, let us arbitrarily throw away the largest and smallest values of each parameter. Then, for each range of values observed, we can calculate the number of expected pairs of programs with equal values for that given parameter. Let us begin with TOTAL OPERS. Range = 151 - 67 + 1 = 85. Any two programs written independently will be assumed to have a total operator count of between 67 and 151, and the probability of them having any one of the possible values is 1/85 * 1/85 = 1/7225. The probability of their having any of the possible values over the entire range is 1/7225 + 1/7225 + . . . + 1/7225 = 1/85. Given that there are 31 input programs, and hence (31 * 30)/2 = 465 pairs of programs, one can expect 5.5, or approximately six pairs of programs to have equal values. We observe four. Following this through, we can calculate expected versus observed pairs for every parameter:

TOTAL OPERS<br>
expected = 465/85 = 5.5 = 6<br>
observed = 4

TOTAL OPNDS<br>
expected = 465/62 = 7.5 = 8<br>
observed = 11

UNIQ OPERS<br>
expected = 465/7 = 66.4 = 67<br>
observed = 73

UNIQ OPNDS<br>
expected = 465/24 = 19.4 = 20<br>
observed = 24

CODE LINES
expected = 465/45 = 10.3 = 11
observed = 19

DECL VARS
expected = 465/24 = 18.6 = 19
observed = 21

FOR STMTS
expected = 465/6 = 77.5 = 78
observed = 104

From the results, it appears not to be unreasonable to assume that all values are equally likely. A statistician, then, can calculate these values and make a judgement as to whether it appears that plagiarism occurred for any parameter. Doing this for every parameter would allow one to conjecture if plagiarism occurred over all parameters and hence over an entire program. Coming up with some final probability that plagiarism occurred for the input programs would contribute to the successful use of Accuse.

## Side Issues

One of the most revealing aspects of this research has been the often enormous variations in the measured parameters. It is incredible to think that two programs as analyzed by Accuse could possibly solve the same problem. This gives rise to a suggested alternate use of Accuse.

Accuse, modified appropriately, could measure the "goodness" of a program. Its analysis could identify both excesses (for example, the programmer used an excessive number of variables) and shortcomings (for example, the programmer used few comments). Accuse is also capable of identifying variables declared and not used. This information could allow a grader to make a quantitative analysis of any program at a glance and grade the program accordingly.

# CHAPTER VIII

## CONCLUSION

The sabotaged programs given as input to Accuse show that it cannot stand alone as a detector of plagiarism, but must in fact be part of a larger system. This system should be one that retrieves the student's program, compiles it, runs it on data the student has never seen, and then sends the student's program into a file that will eventually be processed through Accuse.

Accuse accomplished its goal of being inexpensive to use. Results were actually better than expected.

Finally, Accuse needs to be put into production use to verify or reject assertions made here.

## SELECTED BIBLIOGRAPHY

1. Bulut, N., "Invariant Properties of Algorithms," PhD Thesis, Purdue University (August 1973), 118-119.

2. Fitzsimmon, A.; Love, T., "A Review and Evaluation of Software Science," ACM Computing Surveys, Mar 78, Vol. 10, No. 1.

3. Halstead, M.H., "Elements of Software Science," Elsevier North Holland, New York (1977), Chapters 1-4, 7.

4. Ottenstein, K.J., "An Algorithmic Approach to the Detection and Prevention of Plagiarism," SIGCSE Bulletin, Dec 76, Vol. 8, No. 4.

5. Shaw, M.; Jones, A.; Knueven, P.; McDermott, J.; Miller, P.; Notkin, D., "Cheating Policy in a Computer Science Department," SIGCSE Bulletin, Jul 80, Vol. 12, No. 2.

6. Tukey, John W., "Exploratory Data Analysis," Addison-Wesley Publishing Co., Inc., Philippines (1977) 32-47.

7. Discussions with Dr. Jeff Haemer, Nov 80.

# APPENDIX A

## DEFINITION OF PARAMETERS

1. total lines: total lines in a program.

2. code lines: lines of executable code within a program.

3. code comment lines: lines of comments (excluding declarations) in a program.

4. multiple statement lines: lines of executable code that contain more than one statement.

5. constants and types: number of type and constant declarations.

6. variables declared (and used): variables declared and subsequently used in a program.

7. variables declared (*and not* used): variables declared and subsequently not used in a program.

8. procedures and functions: number of procedures and functions declared in a program.

9. var parameters: *number of var parameters declared and subsequently used in* a program.

10. value parameters: number of parameters declared and subsequently used in a program.

11. procedure variables: number of variables declared (including 9 and 10 above) and subsequently used in a program.

12. for statements: number of for statements in a program.

13. repeat statements: number of repeat statements in a program.

14. while statements: number of while statements in a program.

15. goto statements: number of goto statements in a program.

16. **unique operators:** number of different operators in a program; presently included are: ABS, AND, ARCCOS, ARCSIN, ARCTAN, ARCTAN2, CARD, CASE, CHR, CLOCK, COS, COSH, DATE, DISPOSE, DIV, ELSE, EOF, EOFS, EOLN, EOLNS, EOS, EOSS, EXP, EXPO, FOR, GENSORT, GET, GETSEG, GETRAN, GETRANDOM, GOTO, HALT, HIGH, IF, IN, LINELIMIT, LOG, LOG10, LOW, LN, MESSAGE, MOD, NEW, NOT, OCT, ODD, OR, ORD, PACK, PAGE, POWER, POWERI, PRED, PUT, PUTSEG, RAN, RANDOM, READ, READLN, READSTRING, RELEASE, REPEAT, RESET, REWRITE, ROUND, SETRAN, SETRANDOM, SKIPBLANKS, SIN, SINH, SQR, SQRT, SUCC, TAN, TANH, TIME, TRUNC, UNPACK, UNDEFINED, WHILE, WITH, WRITE, WRITELN, "+", "-", "*", "/", "=", ",", "..", ".", "[", ":=", "<", ">", "<>", "<=", ">=", "^.", and "^".

17. **unique operands:** number of different numbers and variables used in a program.

18. **total operators:** total number of occurrences of the items in 16.

19. **total operands:** total number of occurrences of the items in 17.

20. **indenting function:** ((left indentations) mod 1000) * 1000000 +
(right indentations) mod 1000) * 1000 +
(zero indentations) mod 1000)

# APPENDIX B

## FIRST LISTING

| | TOTAL LINES | CODE LINES | CODE COMNT LINES | MULT STMT LINES | CONS AND TYPES | DECL VARS | VARS NOT USED | PROCS AND FUNCS | VAR PARAM | VAL PARAM | PROC VARS | FOR STMTS | REP STMTS | WHILE STMTS | GOTO STMTS | UNIQ OPERS | UNIQ OPNDS | TOTAL OPERS | TOTAL OPNDS | [illegible] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T110 | 161 | 48 | 48 | 0 | 3 | 12 | 0 | 2 | 3 | 2 | 8 | 0 | 0 | 5 | 0 | 25 | 14 | 90 | 68 | [illegible] |
| T113 | 168 | 72 | 39 | 0 | 3 | 12 | 0 | 2 | 3 | 2 | 8 | 0 | 0 | 4 | 0 | 25 | 14 | 156 | 136 | [illegible] |
| T106 | 158 | 58 | 63 | 0 | 3 | 11 | 0 | 2 | 3 | 2 | 8 | 0 | 0 | 4 | 0 | 25 | 13 | 115 | 96 | [illegible] |
| T103 | 158 | 58 | 63 | 0 | 3 | 11 | 0 | 2 | 3 | 2 | 8 | 0 | 0 | 4 | 0 | 25 | 13 | 115 | 96 | [illegible] |
| T112 | 125 | 71 | 43 | 0 | 3 | 10 | 0 | 2 | 3 | 2 | 6 | 0 | 0 | 7 | 0 | 24 | 12 | 100 | 73 | [illegible] |
| T111 | 174 | 66 | 60 | 0 | 3 | 11 | 0 | 2 | 3 | 2 | 8 | 0 | 0 | 4 | 0 | 24 | 14 | 134 | 117 | [illegible] |
| T109 | 189 | 44 | 70 | 0 | 3 | 11 | 0 | 2 | 3 | 2 | 8 | 0 | 0 | 3 | 0 | 24 | 13 | 97 | 81 | [illegible] |
| T101 | 178 | 56 | 64 | 0 | 3 | 11 | 0 | 2 | 3 | 2 | 8 | 0 | 0 | 4 | 0 | 25 | 13 | 117 | 98 | [illegible] |
| T108 | 226 | 79 | 80 | 0 | 3 | 11 | 0 | 2 | 3 | 2 | 8 | 0 | 0 | 4 | 0 | 25 | 13 | 183 | 159 | [illegible] |
| T102 | 200 | 64 | 75 | 0 | 3 | 11 | 0 | 2 | 3 | 2 | 8 | 0 | 0 | 4 | 0 | 24 | 13 | 110 | 90 | [illegible] |
| T105 | 200 | 64 | 75 | 0 | 3 | 11 | 0 | 2 | 3 | 2 | 8 | 0 | 0 | 4 | 0 | 24 | 13 | 110 | 90 | [illegible] |
| T104 | 189 | 53 | 78 | 0 | 3 | 11 | 0 | 2 | 3 | 2 | 8 | 0 | 0 | 4 | 0 | 25 | 13 | 118 | 106 | [illegible] |
| T107 | 189 | 63 | 78 | 0 | 3 | 11 | 0 | 2 | 3 | 2 | 8 | 0 | 0 | 4 | 0 | 25 | 13 | 118 | 106 | [illegible] |

* VARIABLE(S) WERE DECLARED BUT NOT USED

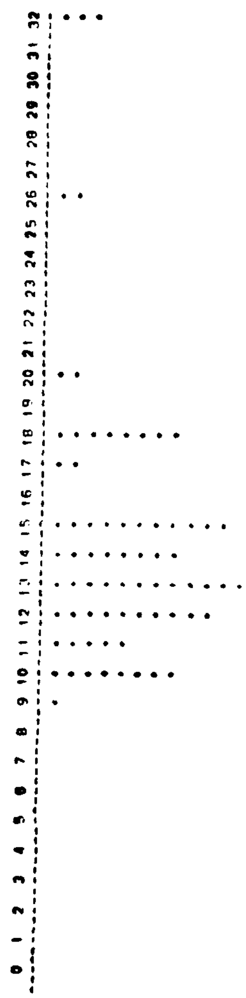
** PROCEDURE(S) WERE DECLARED BUT NOT USED

*** VARIABLE(S) AND PROCEDURE(S) WERE DECLARED BUT NOT USED

| TOTAL OPERS | | TOTAL OPNDS | | UNIQ OPERS | | UNIQ OPNDS | | CODE LINES | | DECL VARS | | FOR STM'S | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T110 | 90 | T110 | 68 | T102 | 24 | T112 | 12 | T109 | 44 | T112 | 10 | T109 | 3 |
| T109 | 97 | T112 | 73 | T105 | 24 | T107 | 13 | T110 | 48 | T102 | 11 | T106 | 4 |
| T112 | 100 | T109 | 81 | T111 | 24 | T108 | 13 | T101 | 56 | T107 | 11 | T101 | 4 |
| T105 | 110 | T102 | 90 | T112 | 24 | T103 | 13 | T106 | 58 | T105 | 11 | T113 | 4 |
| T102 | 110 | T105 | 90 | T109 | 24 | T101 | 13 | T103 | 58 | T108 | 11 | T103 | 4 |
| T103 | 115 | T106 | 96 | T104 | 25 | T106 | 13 | T104 | 63 | T111 | 11 | T104 | 4 |
| T106 | 115 | T103 | 96 | T108 | 25 | T109 | 13 | T107 | 63 | T101 | 11 | T107 | 4 |
| T101 | 117 | T101 | 98 | T113 | 25 | T105 | 13 | T102 | 64 | T109 | 11 | T102 | 4 |
| T104 | 118 | T107 | 106 | T107 | 25 | T104 | 13 | T105 | 64 | T106 | 11 | T105 | 4 |
| T107 | 118 | T104 | 106 | T106 | 25 | T102 | 13 | T111 | 66 | T104 | 11 | T111 | 4 |
| T111 | 134 | T111 | 117 | T110 | 25 | T110 | 14 | T112 | 71 | T103 | 11 | T108 | 4 |
| T113 | 156 | T113 | 136 | T101 | 25 | T113 | 14 | T113 | 72 | T113 | 12 | T110 | 5 |
| T108 | 183 | T108 | 159 | T103 | 25 | T111 | 14 | T108 | 79 | T110 | 12 | T112 | 7 |

WARNING: HEADING(S) ARE NOT CORRECT. AT LEAST ONE REFLECTS A SUM OF TWO COUNTS.

| | T102 | T103 | T104 | T105 | T106 | T107 | T108 | T109 | T110 | T111 | T112 | T113 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 101 | 14 | 28 | 20 | 14 | 26 | 20 | 15 | 13 | 12 | 13 | 10 | 13 |
| 102 | | 15 | 18 | 32 | 15 | 18 | 14 | 14 | 11 | 17 | 11 | 12 |
| 103 | | | 18 | 15 | 32 | 18 | 15 | 13 | 12 | 13 | 10 | 13 |
| 104 | | | | 18 | 18 | 32 | 15 | 13 | 12 | 15 | 10 | 13 |
| 105 | | | | | 15 | 18 | 14 | 14 | 11 | 17 | 11 | 12 |
| 106 | | | | | | 18 | 15 | 13 | 12 | 13 | 10 | 13 |
| 107 | | | | | | | 15 | 13 | 12 | 15 | 10 | 13 |
| 108 | | | | | | | | 13 | 12 | 13 | 10 | 13 |
| 109 | | | | | | | | | 10 | 13 | 14 | 11 |
| 110 | | | | | | | | | | 12 | 9 | 14 |
| 111 | | | | | | | | | | | 10 | 13 |
| 112 | | | | | | | | | | | | 12 |




FREQUENCY DISTRIBUTION GRAPH FOR PAIRS OF PROGRAMS

```
THE FOLLOWING PAIRS HAVE A CORRELATION OF 32:
        T102,T105
        T103,T108
        T104,T107
```

# APPENDIX C

## SECOND LISTING

| | TOTAL LINES | CODE LINES | CODE COMNT LINES | MULT STMT LINES | CONS AND TYPES | DECL VARS | VARS NOT USED | PROCS AND FUNCS | VAR PARAM | VAL PARAM | PROC VARS | FOR STMTS | REP STMTS | WHILE STMTS | GOTO STMTS | UNIQ OPERS | UNIQ OPNDS | TOTAL OPERS | TOTAL OPNDS | INDENTING FUNCTION |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DPG4 | 70 | 49 | 6 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 20 | 12 | 63 | 57 | 54 |
| DCB3 | 70 | 30 | 10 | 0 | 0 | 8 | 1 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 21 | 14 | 54 | 46 | [illegible]38 * |
| DD75 | 103 | 36 | 24 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 22 | 11 | 62 | 50 | [illegible]41 |
| DD50 | 73 | 41 | 2 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 21 | 12 | 56 | 46 | [illegible]46 |
| DD14 | 103 | 44 | 16 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 22 | 13 | 62 | 50 | [illegible]48 |
| DJ18 | 59 | 36 | 11 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 21 | 12 | 55 | 43 | [illegible]40 |
| DA47 | 100 | 58 | 16 | 0 | 1 | 10 | 1 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 19 | 16 | 58 | 39 | [illegible]71 * |
| DC67 | 36 | 20 | 4 | 0 | 1 | 5 | 2 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 10 | 19 | 21 | 31 | [illegible]18 * |
| DC42 | 80 | 29 | 16 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 23 | 12 | 65 | 61 | [illegible]76 |
| DM55 | 98 | 43 | 22 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 21 | 12 | 57 | 50 | [illegible]44 |
| DH29 | 69 | 40 | 22 | 0 | 2 | 7 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 21 | 12 | 54 | 48 | [illegible]44 |
| DD05 | 78 | 43 | 11 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 1 | 0 | 20 | 14 | 60 | 60 | [illegible]50 |
| DC01 | 104 | 41 | 18 | 0 | 8 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 22 | 10 | 81 | 51 | [illegible]13 |
| DD13 | 83 | 43 | 13 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 23 | 15 | 73 | 70 | [illegible]45 |
| DC43 | 83 | 48 | 13 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 25 | 13 | 75 | 68 | [illegible]49 |
| DC53 | 92 | 51 | 17 | 0 | 9 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 22 | 11 | 68 | 58 | [illegible]53 |
| E103 | 87 | 54 | 23 | 1 | 0 | 11 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 5 | 22 | 18 | 62 | 60 | [illegible]59 |
| DD27 | 85 | 37 | 32 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 22 | 15 | 64 | 55 | [illegible]37 |
| DC64 | 88 | 36 | 18 | 0 | 3 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 21 | 12 | 53 | 49 | [illegible]38 |
| DD51 | 84 | 35 | 10 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 23 | 15 | 67 | 58 | [illegible]10 |
| DC73 | 93 | 46 | 16 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 21 | 13 | 62 | 55 | [illegible]16 |
| DD12 | 81 | 42 | 17 | 0 | 1 | 11 | 1 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 19 | 16 | 50 | 44 | [illegible]47 * |
| DC46 | 120 | 50 | 19 | 0 | 2 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 21 | 12 | 64 | 56 | [illegible]53 |
| DC53 | 90 | 45 | 21 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 23 | 13 | 64 | 55 | [illegible]45 |
| DC72 | 85 | 50 | 16 | 0 | 2 | 5 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 3 | 0 | 19 | 8 | 61 | 57 | [illegible]52 |
| DC79 | 108 | 52 | 18 | 0 | 4 | 10 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 1 | 0 | 22 | 14 | 58 | 51 | [illegible]54 |
| DJ37 | 90 | 56 | 13 | 0 | 2 | 9 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 2 | 0 | 18 | 13 | 55 | 52 | [illegible]57 |
| DD01 | 96 | 60 | 15 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 20 | 12 | 64 | 55 | [illegible]60 |
| DD28 | 113 | 61 | 15 | 0 | 3 | 9 | 0 | 0 | 0 | 0 | 0 | 6 | 1 | 0 | 0 | 22 | 13 | 93 | 92 | [illegible] |
| DD71 | 109 | 55 | 22 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 21 | 14 | 60 | 55 | [illegible]16 |
| DD96 | 74 | 61 | 6 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 23 | 15 | 81 | 63 | [illegible] |
| DD07 | 128 | 58 | 17 | 0 | 0 | 11 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 21 | 15 | 86 | 74 | [illegible]62 |
| DD26 | 117 | 73 | 17 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 22 | 17 | 92 | 81 | [illegible]23 |
| DD68 | 51 | 39 | 4 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 24 | 13 | 55 | 47 | [illegible]47 |
| C132 | 91 | 44 | 16 | 0 | 0 | 9 | 2 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 22 | 15 | 57 | 57 | [illegible]42 * |
| DC81 | 63 | 37 | 8 | 0 | 2 | 8 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 1 | 0 | 20 | 13 | 45 | 42 | [illegible]37 |
| DD19 | 78 | 39 | 19 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 19 | 16 | 48 | 44 | [illegible]31 |
| 2278 | 83 | 43 | 13 | 0 | 4 | 6 | 1 | 0 | 0 | 0 | 0 | 2 | 0 | 3 | 0 | 21 | 9 | 57 | 45 | [illegible]12 * |
| DD10 | 78 | 44 | 12 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 20 | 13 | 57 | 46 | [illegible]43 |
| DD79 | 102 | 45 | 15 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 20 | 13 | 60 | 59 | [illegible]47 |
| DD84 | 118 | 46 | 11 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 22 | 15 | 65 | 63 | [illegible]18 |
| DD23 | 114 | 51 | 25 | 0 | 3 | 6 | 0 | 0 | 0 | 0 | 0 | 5 | 1 | 0 | 0 | 24 | 10 | 68 | 67 | [illegible]19 |
| DL05 | 90 | 57 | 16 | 0 | 1 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 21 | 12 | 64 | 46 | [illegible]55 |
| DD78 | 118 | 61 | 11 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 7 | 0 | 1 | 0 | 21 | 14 | 78 | 65 | [illegible]65 |
| DD47 | 104 | 45 | 45 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 20 | 15 | 59 | 50 | [illegible]10 |
| DD64 | 83 | 54 | 6 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 25 | 12 | 79 | 75 | [illegible]53 |
| DD37 | 74 | 36 | 14 | 0 | 2 | 6 | 0 | 0 | 0 | 0 | 0 | 5 | 1 | 0 | 0 | 19 | 9 | 54 | 45 | [illegible]35 |
| DD17 | 87 | 44 | 3 | 1 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 21 | 12 | 57 | 46 | [illegible]10 |
| DD67 | 63 | 48 | 3 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 20 | 11 | 61 | 56 | [illegible]43 |
| DD41 | 96 | 58 | 17 | 0 | 2 | 6 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 24 | 9 | 68 | 53 | [illegible]55 |

| | TOTAL LINES | CODE LINES | CODE COMNT LINES | MULT STMT LINES | CONS AND TYPES | DECL VARS | VARS NOT USED | PROCS AND FUNCS | VAR PARAM | VAL PARAM | PROC VARS | FOR STMTS | REP STMTS | WHILE STMTS | GOTO STMTS | UNIQ OPERS | UNIQ OPNDS | TOTAL OPERS | TOTAL OPNDS | [illegible] FUNCTION |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DD07 | 91 | 42 | 23 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 2 | 0 | 23 | 12 | 59 | 50 | [illegible] |
| DD85 | 103 | 44 | 19 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 21 | 12 | 57 | 50 | [illegible] |
| AY70 | 81 | 47 | 14 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 21 | 14 | 61 | 60 | [illegible] |
| GD18 | 101 | 67 | 4 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 1 | 0 | 21 | 15 | 87 | 79 | [illegible] |
| 1418 | 74 | 39 | 13 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 22 | 12 | 61 | 53 | [illegible] |
| DD04 | 78 | 40 | 12 | 0 | 4 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 23 | 11 | 78 | 63 | [illegible] |
| DD21 | 109 | 54 | 28 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 19 | 16 | 65 | 62 | [illegible] |
| DD09 | 118 | 39 | 28 | 0 | 2 | 6 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 22 | 9 | 56 | 46 | [illegible] |
| D[illegible]18 | 101 | 44 | 14 | 0 | 3 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 20 | 11 | 61 | 60 | [illegible] |
| C145 | 103 | 52 | 33 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 23 | 13 | 77 | 72 | [illegible] |
| DD18 | 92 | 64 | 4 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 3 | 0 | 22 | 16 | 76 | 65 | [illegible] |
| DD03 | 113 | 44 | 34 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 21 | 15 | 67 | 77 | [illegible] |
| DC95 | 100 | 54 | 27 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 7 | 0 | 1 | 0 | 23 | 10 | 80 | 74 | [illegible] |
| DCH5 | 93 | 54 | 9 | 0 | 2 | 10 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 1 | 0 | 22 | 13 | 58 | 50 | [illegible] |
| PY46 | 119 | 48 | 44 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 1 | 0 | 21 | 14 | 70 | 62 | [illegible] |
| F101 | 100 | 54 | 30 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 22 | 12 | 63 | 56 | [illegible] |
| DD74 | 87 | 43 | 11 | 3 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 21 | 12 | 58 | 49 | [illegible] |
| DC74 | 81 | 48 | 9 | 0 | 2 | 5 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 20 | 11 | 59 | 51 | [illegible] |
| DC41 | 100 | 51 | 21 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 3 | 1 | 2 | 0 | 21 | 10 | 79 | 70 | [illegible] |
| DD93 | 122 | 97 | 0 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 2 | 0 | 23 | 14 | 107 | 81 | [illegible] |
| DW77 | 88 | 62 | 10 | 6 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 2 | 3 | 0 | 0 | 23 | 15 | 73 | 68 | [illegible] |
| AR28 | 103 | 70 | 11 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 1 | 0 | 25 | 15 | 90 | 78 | [illegible] |
| DC65 | 135 | 50 | 24 | 0 | 12 | 8 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 21 | 11 | 69 | 67 | [illegible] |
| DC92 | 131 | 89 | 9 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 3 | 0 | 20 | 13 | 112 | 111 | [illegible] |
| DD53 | 84 | 51 | 13 | 0 | 1 | 10 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 18 | 17 | 56 | 51 | [illegible] |
| DC38 | 71 | 42 | 29 | 0 | 4 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 21 | 10 | 61 | 54 | [illegible] |
| DJ91 | 90 | 60 | 7 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 1 | 0 | 24 | 18 | 83 | 74 | [illegible] |
| DD52 | 112 | 60 | 15 | 0 | 3 | 8 | 1 | 0 | 0 | 0 | 0 | 8 | 0 | 1 | 0 | 20 | 11 | 63 | 59 | [illegible] |
| DC97 | 140 | 63 | 27 | 0 | 0 | 11 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 19 | 16 | 71 | 59 | [illegible] |
| L603 | 104 | 69 | 9 | 0 | 3 | 7 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 1 | 0 | 24 | 11 | 78 | 65 | [illegible] |
| DD16 | 99 | 52 | 18 | 0 | 2 | 5 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 4 | 0 | 25 | 9 | 98 | 73 | [illegible] |
| DD68 | 96 | 49 | 15 | 1 | 3 | 10 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 23 | 14 | 78 | 71 | [illegible] |
| DC99 | 96 | 69 | 14 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 18 | 13 | 66 | 49 | [illegible] |
| E104 | 87 | 57 | 11 | 0 | 4 | 11 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 3 | 0 | 22 | 17 | 69 | 66 | [illegible] |
| DD07 | 105 | 46 | 24 | 0 | 3 | 8 | 0 | 0 | 0 | 0 | 0 | 7 | 0 | 1 | 0 | 23 | 9 | 68 | 66 | [illegible] |
| DJ75 | 104 | 57 | 34 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 20 | 14 | 65 | 53 | [illegible] |
| RW19 | 136 | 39 | 45 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 22 | 13 | 58 | 46 | [illegible] |
| DC39 | 81 | 55 | 17 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 1 | 0 | 24 | 14 | 75 | 63 | [illegible] |
| R[illegible]40 | 91 | 45 | 22 | 1 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 22 | 16 | 62 | 60 | [illegible] |
| DF72 | 87 | 50 | 10 | 0 | 2 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 2 | 0 | 23 | 10 | 68 | 58 | [illegible] |
| DC[illegible]6 | 97 | 52 | 22 | 0 | 3 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 24 | 12 | 64 | 59 | [illegible] |
| DD77 | 103 | 51 | 39 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 24 | 15 | 68 | 61 | [illegible] |
| CF62 | 85 | 51 | 9 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 2 | 0 | 23 | 15 | 63 | 58 | [illegible] |
| D189 | 149 | 90 | 10 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 7 | 0 | 1 | 0 | 21 | 14 | 93 | 81 | [illegible] |
| DC51 | 70 | 50 | 16 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 20 | 14 | 56 | 52 | [illegible] |
| D[illegible]45 | 113 | 62 | 17 | 0 | 2 | 7 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 2 | 0 | 18 | 10 | 65 | 57 | [illegible] |
| E102 | 95 | 66 | 27 | 1 | 2 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 21 | 15 | 69 | 57 | [illegible] |
| DJ74 | 78 | 41 | 22 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 23 | 14 | 67 | 61 | [illegible] |
| DJ75 | 93 | 58 | 25 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 7 | 0 | 1 | 0 | 24 | 13 | 77 | 71 | [illegible] |
| DM67 | 74 | 40 | 7 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 23 | 11 | 59 | 48 | [illegible] |

| | TOTAL LINES | CODE LINES | CODE COMNT LINES | MULT STMT LINES | CONS AND TYPES | DECL VARS | VARS NOT USED | PROCS AND FUNCS | VAR PARAM | VAL PARAM | PROC VARS | FOR STMTS | REP STMTS | WHILE STMTS | GOTO STMTS | UNIQ OPERS | UNIQ OPNDS | TOTAL OPERS | TOTAL OPNDS | ENDING [illegible] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DCR6 | 115 | 73 | 58 | 0 | 3 | 7 | 0 | 0 | 0 | 0 | 0 | 3 | 1 | 3 | 0 | 24 | 12 | 89 | 78 | [illegible] |
| DD11 | 113 | 53 | 8 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 24 | 15 | 78 | 78 | [illegible] |
| DD13 | 133 | 72 | 40 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 1 | 0 | 22 | 14 | 88 | 88 | [illegible] |
| DK22 | 74 | 42 | 2 | 0 | 0 | 7 | 2 | 0 | 0 | 0 | 0 | 6 | 0 | 1 | 0 | 21 | 13 | 55 | 49 | [illegible] * |
| DC55 | 108 | 43 | 17 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 1 | 0 | 21 | 15 | 82 | 88 | [illegible] |
| DD68 | 118 | 62 | 15 | 0 | 6 | 6 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 22 | 10 | 72 | 66 | [illegible] |
| DM17 | 97 | 42 | 32 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 23 | 14 | 55 | 46 | [illegible] |
| [illegible] | 83 | 51 | 21 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 20 | 14 | 49 | 37 | [illegible] |
| [illegible] | 77 | 46 | 19 | 0 | 5 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 19 | 10 | 60 | 48 | [illegible] |
| DC14 | 118 | 58 | 48 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 23 | 12 | 81 | 63 | [illegible] |
| DD78 | 66 | 33 | 9 | 0 | 2 | 8 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 21 | 14 | 56 | 46 | [illegible] |
| DC89 | 61 | 41 | 8 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 21 | 12 | 51 | 37 | [illegible] |
| DD28 | 101 | 81 | 28 | 0 | 2 | 10 | 0 | 0 | 0 | 0 | 0 | 5 | 1 | 1 | 0 | 28 | 16 | 86 | 84 | [illegible] |
| [illegible] | 108 | 78 | 15 | 0 | 2 | 8 | 0 | 0 | 0 | 0 | 0 | 6 | 2 | 1 | 0 | 26 | 11 | 79 | 60 | [illegible] |
| DD22 | 109 | 54 | 10 | 0 | 3 | 7 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 1 | 0 | 21 | 10 | 67 | 58 | [illegible] |
| [illegible] | 175 | 90 | 43 | 0 | 5 | 8 | 0 | 0 | 0 | 0 | 0 | 2 | 3 | 1 | 0 | 24 | 16 | 123 | 118 | [illegible] |
| CC20 | 153 | 51 | 42 | 0 | 3 | 6 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 25 | 9 | 78 | 71 | [illegible] |
| [illegible] | 110 | 54 | 6 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 23 | 14 | 62 | 50 | [illegible] |
| DD38 | 165 | 88 | 39 | 0 | 2 | 5 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 1 | 0 | 24 | 16 | 121 | 121 | [illegible] |
| V131 | 57 | 45 | 3 | 0 | 2 | 5 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 19 | 10 | 52 | 49 | [illegible] |
| DD81 | 68 | 43 | 20 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 23 | 13 | 61 | 55 | [illegible] |
| [illegible] | 78 | 44 | 15 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 20 | 12 | 70 | 52 | [illegible] |
| [illegible] | 103 | 46 | 13 | 0 | 3 | 7 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 21 | 11 | 54 | 48 | [illegible] |
| [illegible] | 95 | 43 | 15 | 0 | 4 | 7 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 24 | 11 | 57 | 48 | [illegible] |
| [illegible] | 107 | 50 | 24 | 0 | 3 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 21 | 14 | 62 | 62 | [illegible] |
| [illegible] | 137 | 62 | 30 | 0 | 4 | 7 | 1 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 19 | 11 | 78 | 67 | [illegible] * |
| DC16 | 111 | 54 | 22 | 0 | 2 | 8 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 21 | 9 | 64 | 49 | [illegible] |
| [illegible] | 79 | 51 | 5 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 2 | 0 | 16 | 13 | 55 | 44 | [illegible] |
| [illegible] | 146 | 65 | 27 | 0 | 5 | 7 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 3 | 0 | 23 | 10 | 82 | 70 | [illegible] |
| [illegible] | 84 | 54 | 6 | 0 | 2 | 11 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 21 | 17 | 58 | 47 | [illegible] |
| [illegible] | 91 | 46 | 16 | 0 | 0 | 7 | 1 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 22 | 15 | 59 | 59 | [illegible] * |
| C063 | 93 | 54 | 8 | 0 | 2 | 6 | 1 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 21 | 11 | 62 | 54 | [illegible] * |
| CD32 | 102 | 75 | 10 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 2 | 0 | 22 | 14 | 84 | 68 | [illegible] |
| DD33 | 105 | 62 | 16 | 0 | 1 | 10 | 1 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 17 | 17 | 64 | 52 | [illegible] * |
| DDH2 | 91 | 55 | 4 | 0 | 2 | 14 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 1 | 0 | 23 | 19 | 81 | 50 | [illegible] |
| DP06 | 171 | 62 | 31 | 0 | 2 | 7 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 21 | 11 | 81 | 68 | [illegible] |
| DC93 | 145 | 66 | 23 | 2 | 1 | 10 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 21 | 10 | 71 | 61 | [illegible] |

* VARIABLE(S) WERE DECLARED BUT NOT USED

** PROCEDURE(S) WERE DECLARED BUT NOT USED

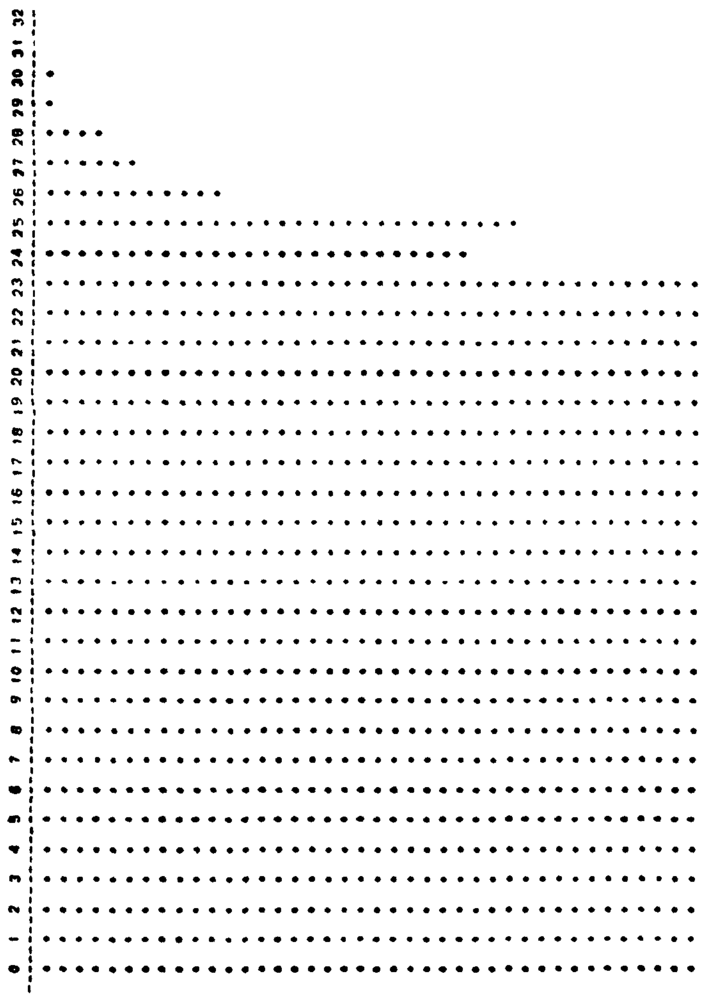
*** VARIABLE(S) AND PROCEDURE(S) WERE DECLARED BUT NOT USED

| TOTAL OPERS | | TOTAL OPNDS | | UNIQ OPERS | | UNIQ OPNDS | | CODE LINES | | DECL VARS | | FOR STMTS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DC57 | 21 | DC67 | 31 | DC67 | 10 | DC72 | 8 | DC67 | 20 | DC41 | 5 | DC67 | 2 |
| DC01 | 45 | DD46 | 37 | DJ45 | 16 | DD41 | 9 | DC42 | 29 | DD87 | 5 | DC19 | 3 |
| DD39 | 48 | DC89 | 37 | DD33 | 17 | DD37 | 9 | DC83 | 30 | DC73 | 5 | DC55 | 4 |
| DD46 | 49 | DA47 | 39 | DC45 | 18 | DD15 | 9 | DD76 | 33 | V191 | 5 | 2278 | 5 |
| DD12 | 50 | DC81 | 42 | DD53 | 18 | DC36 | 9 | DD51 | 36 | DC95 | 5 | DM77 | 5 |
| DC89 | 51 | DJ48 | 43 | DJ87 | 18 | DD93 | 9 | DJ48 | 36 | DC74 | 5 | DP18 | 5 |
| V191 | 52 | DJ45 | 44 | DC99 | 18 | DD07 | 9 | DC64 | 36 | DD85 | 5 | DD56 | 5 |
| DH99 | 54 | DD39 | 44 | V191 | 19 | 2278 | 9 | DD75 | 36 | DC64 | 5 | DD44 | 5 |
| DC83 | 54 | DD12 | 44 | DC72 | 19 | CC20 | 9 | DD37 | 36 | BM55 | 5 | DD17 | 5 |
| DD37 | 54 | DD37 | 45 | DD21 | 19 | DC41 | 10 | DC81 | 37 | DD67 | 5 | DD76 | 5 |
| HIST | 54 | 2278 | 45 | Y265 | 19 | DC95 | 10 | DD27 | 37 | DC72 | 5 | DD87 | 5 |
| DC68 | 55 | DP18 | 46 | DG35 | 19 | CC26 | 10 | DD39 | 39 | DD16 | 6 | DC65 | 5 |
| DM47 | 55 | DD09 | 46 | DA47 | 19 | DD22 | 10 | 1418 | 39 | DC67 | 5 | E101 | 5 |
| DJ07 | 55 | BW19 | 46 | DC97 | 19 | 2F72 | 10 | DD09 | 39 | DD36 | 5 | DJ17 | 5 |
| DJ48 | 55 | DD17 | 45 | DD12 | 19 | DC38 | 10 | BW19 | 39 | DD74 | 6 | E104 | 5 |
| DJ45 | 55 | DC05 | 46 | DD37 | 19 | DC91 | 10 | DC68 | 39 | DD41 | 6 | DD75 | 5 |
| DK22 | 55 | DD76 | 46 | DD39 | 19 | DC45 | 10 | DD04 | 40 | DD54 | 6 | DD51 | 5 |
| DD58 | 56 | DD58 | 46 | DD52 | 20 | V191 | 10 | DH09 | 40 | DC36 | 6 | DM05 | 5 |
| DD09 | 56 | DC83 | 46 | DP64 | 20 | DD23 | 10 | DM67 | 40 | DD63 | 6 | DD58 | 5 |
| DD74 | 56 | DM47 | 46 | DD47 | 20 | DG35 | 10 | DC89 | 41 | DD17 | 6 | DA47 | 5 |
| BW19 | 56 | DC68 | 47 | DD61 | 20 | DF66 | 10 | DC58 | 41 | DJ75 | 6 | DD85 | 5 |
| DC51 | 56 | DD62 | 47 | DC51 | 20 | DC53 | 11 | DC91 | 41 | DD61 | 6 | F103 | 5 |
| DD53 | 56 | HIST | 48 | DC75 | 20 | DC28 | 11 | DJ74 | 41 | DD66 | 6 | DD68 | 5 |
| DD76 | 56 | DM67 | 48 | DC74 | 20 | DC63 | 11 | DD12 | 42 | DD07 | 6 | DH99 | 5 |
| DD56 | 57 | DG35 | 48 | DD67 | 20 | DD75 | 11 | DK22 | 42 | 2278 | 6 | DD13 | 5 |
| C182 | 57 | DH09 | 48 | DD79 | 20 | DD94 | 11 | DC38 | 42 | DD52 | 6 | HIST | 5 |
| DD85 | 57 | DD56 | 48 | DD01 | 20 | DP06 | 11 | DM47 | 42 | DD77 | 6 | DC68 | 5 |
| DD17 | 57 | DD74 | 49 | DW85 | 20 | HIST | 11 | DD87 | 42 | C145 | 6 | DD27 | 5 |
| 2278 | 57 | DC36 | 49 | DC82 | 20 | DC65 | 11 | DC55 | 43 | DM67 | 6 | DD48 | 5 |
| DP18 | 57 | DC64 | 49 | DC81 | 20 | DM67 | 11 | DD13 | 43 | DD09 | 6 | DD74 | 5 |
| BM55 | 57 | DK22 | 49 | DP18 | 20 | DC52 | 11 | DD74 | 43 | DG35 | 6 | DC99 | 6 |
| DC79 | 58 | V191 | 49 | DD46 | 20 | DD56 | 11 | DD81 | 43 | DD01 | 6 | DC71 | 6 |
| DC85 | 58 | DC79 | 49 | DC98 | 20 | DC74 | 11 | BM55 | 43 | 1410 | 6 | DD04 | 6 |
| DD52 | 58 | DD14 | 50 | DD18 | 21 | DD67 | 11 | DM85 | 43 | DD23 | 6 | DC98 | 6 |
| DC64 | 58 | DD47 | 50 | DC73 | 21 | LF93 | 11 | DD56 | 43 | DD75 | 6 | DD93 | 6 |
| DA47 | 58 | DD75 | 50 | DC71 | 21 | DD52 | 11 | 2278 | 43 | CC26 | 6 | BY48 | 6 |
| DD87 | 59 | DD82 | 50 | DW02 | 21 | Y265 | 11 | DP18 | 44 | DD43 | 6 | DD01 | 6 |
| DC72 | 59 | DC49 | 50 | DD63 | 21 | DJ85 | 12 | DD17 | 44 | DD18 | 6 | DP06 | 6 |
| DC74 | 59 | DC85 | 50 | DC38 | 21 | DC61 | 12 | DD44 | 44 | DD37 | 6 | Y265 | 6 |
| DD47 | 59 | DD85 | 50 | DJ89 | 21 | DC66 | 12 | DC98 | 44 | DC92 | 7 | DD13 | 6 |
| DM67 | 59 | DE87 | 50 | DC55 | 21 | DP64 | 12 | C182 | 44 | DC14 | 7 | F102 | 6 |
| DG35 | 60 | BM55 | 50 | C425 | 21 | DC64 | 12 | DD61 | 44 | DC49 | 7 | DC75 | 6 |
| DC71 | 60 | DC91 | 51 | BY46 | 21 | DC95 | 12 | DD85 | 44 | DD46 | 7 | 2276 | 6 |
| DM85 | 60 | DC79 | 51 | DD70 | 21 | DD17 | 12 | DD03 | 44 | DC53 | 7 | DD21 | 6 |
| DD79 | 60 | DD53 | 51 | AY70 | 21 | DD58 | 12 | DC79 | 45 | DD22 | 7 | DD53 | 6 |
| AY70 | 61 | DC74 | 51 | DD22 | 21 | DC93 | 12 | DC63 | 45 | E101 | 7 | DD39 | 6 |
| 1418 | 61 | DJ87 | 52 | DC93 | 21 | BM55 | 12 | DD47 | 45 | DC66 | 7 | DC96 | 5 |
| DC98 | 61 | DD33 | 52 | DC41 | 21 | DH39 | 12 | BY48 | 45 | DD95 | 7 | DD11 | 6 |
| DC38 | 61 | DD61 | 52 | DC46 | 21 | DD74 | 12 | V191 | 45 | BF72 | 7 | DJ87 | 6 |
| DD81 | 61 | DC51 | 52 | DD03 | 21 | DJ48 | 12 | DC84 | 46 | DC43 | 7 | DD08 | 6 |

| TOTAL OPERS | | TOTAL OPNDS | | UNIQ OPERS | | UNIQ OPNDS | | CODE LINES | | DECL VARS | | FOR STMTS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DC91 | 61 | DC75 | 53 | DC65 | 21 | DD64 | 12 | DG35 | 46 | DC39 | 7 | DN02 | 6 |
| DC72 | 61 | T418 | 53 | E102 | 21 | E101 | 12 | HIST | 46 | DC46 | 7 | DD12 | 6 |
| DD82 | 61 | DD41 | 53 | DP06 | 21 | DD87 | 12 | DD07 | 46 | CA25 | 7 | DC97 | 6 |
| DD67 | 61 | UC30 | 54 | DD85 | 21 | DC86 | 12 | DC92 | 46 | DK22 | 7 | DC93 | 6 |
| BY48 | 62 | DD63 | 54 | DK22 | 21 | D 42 | 12 | DC73 | 46 | DJ74 | 7 | DD62 | 6 |
| DD44 | 62 | DD27 | 55 | DC83 | 21 | T418 | 12 | AY70 | 47 | DC91 | 7 | C102 | 6 |
| DD63 | 62 | DC73 | 55 | DC64 | 21 | DC14 | 12 | DC74 | 48 | DC55 | 7 | DC63 | 6 |
| CA25 | 62 | DC71 | 55 | DH99 | 21 | DC46 | 12 | DD67 | 48 | CS90 | 7 | DD47 | 6 |
| DD75 | 62 | DC92 | 55 | DD62 | 21 | DD01 | 12 | BY46 | 48 | DC38 | 7 | DD79 | 6 |
| DC73 | 62 | DD01 | 55 | HIST | 21 | DP25 | 13 | DC43 | 48 | DC89 | 7 | BW19 | 6 |
| DC49 | 62 | DC63 | 55 | 2278 | 21 | DC99 | 13 | DD68 | 49 | DD27 | 7 | DC51 | 6 |
| E103 | 62 | DD81 | 55 | DJ48 | 21 | DD44 | 13 | DP64 | 49 | DC86 | 7 | AY70 | 6 |
| E101 | 63 | DD79 | 55 | DC38 | 21 | DGH1 | 13 | BP72 | 50 | DJ48 | 7 | DC64 | 6 |
| DD52 | 63 | BP72 | 56 | DD17 | 21 | DJ75 | 13 | DC72 | 50 | DH99 | 7 | DJ45 | 6 |
| CE62 | 63 | DC46 | 56 | DD05 | 21 | DC63 | 13 | DC51 | 50 | DC68 | 7 | DC83 | 6 |
| DP64 | 63 | E101 | 56 | DC89 | 21 | BW19 | 13 | DC65 | 50 | L693 | 7 | DC84 | 6 |
| DC36 | 64 | DD67 | 56 | DD58 | 21 | DJ87 | 13 | CA25 | 50 | HIST | 7 | DC42 | 6 |
| DD33 | 64 | DC72 | 57 | DD74 | 21 | DC81 | 13 | DC46 | 50 | DC45 | 7 | DC49 | 6 |
| DC46 | 64 | E102 | 57 | DD76 | 21 | DC42 | 13 | DJ45 | 51 | DC98 | 7 | C145 | 6 |
| DC66 | 64 | C182 | 57 | BM55 | 21 | DC85 | 13 | CC20 | 51 | DD03 | 7 | DM67 | 6 |
| DD27 | 64 | DC45 | 57 | E103 | 22 | DC68 | 13 | DC41 | 51 | BY49 | 7 | DD09 | 6 |
| DD05 | 64 | DP64 | 57 | DP26 | 22 | DP18 | 13 | CE62 | 51 | Y265 | 7 | DD61 | 6 |
| DD01 | 64 | DC53 | 58 | DD43 | 22 | DC73 | 13 | DD23 | 51 | DP06 | 7 | DD68 | 6 |
| DC63 | 64 | CE62 | 58 | DD09 | 22 | C145 | 13 | DC53 | 51 | DD44 | 7 | DD77 | 6 |
| DC75 | 65 | DD51 | 58 | DD06 | 22 | DC43 | 13 | DD26 | 51 | DD56 | 7 | DD23 | 6 |
| DC45 | 65 | DD22 | 58 | BW19 | 22 | DJ45 | 13 | DD46 | 51 | DDP1 | 7 | CC20 | 6 |
| DD21 | 65 | DC97 | 59 | DD32 | 22 | DD79 | 13 | DD77 | 51 | DD13 | 7 | DD37 | 6 |
| DC42 | 65 | DC66 | 59 | DD48 | 22 | DK22 | 13 | DD53 | 51 | DP18 | 7 | DG35 | 6 |
| DC84 | 65 | DD52 | 59 | DC84 | 22 | DC79 | 14 | DC66 | 52 | 2926 | 8 | DD01 | 6 |
| DC99 | 66 | DC52 | 60 | DD66 | 22 | DD83 | 14 | DD16 | 52 | DC52 | 8 | T418 | 6 |
| DD77 | 66 | DM85 | 60 | E104 | 22 | DM47 | 14 | C145 | 52 | DC82 | 8 | DC74 | 6 |
| DD41 | 66 | BY48 | 60 | BY48 | 22 | DC39 | 14 | DC79 | 52 | DD21 | 8 | DP64 | 6 |
| DD22 | 67 | E103 | 60 | E101 | 22 | DD58 | 14 | DD11 | 53 | E102 | 8 | BM55 | 6 |
| DD03 | 67 | DC98 | 60 | DC92 | 22 | DC49 | 14 | DC36 | 54 | DC75 | 8 | DC41 | 6 |
| DD51 | 67 | AY70 | 60 | C182 | 22 | DJ74 | 14 | E103 | 54 | DC71 | 8 | DC73 | 6 |
| DJ74 | 67 | DC93 | 61 | DC85 | 22 | DD78 | 14 | E101 | 54 | DC99 | 8 | V191 | 6 |
| BP72 | 68 | DD77 | 61 | DD27 | 22 | BY46 | 14 | DD22 | 54 | DD18 | 8 | DD64 | 5 |
| DD07 | 68 | DC42 | 61 | 1410 | 22 | CA25 | 14 | DD64 | 54 | DD04 | 8 | DC38 | 6 |
| DD23 | 68 | DJ74 | 61 | DC91 | 22 | DD76 | 14 | DD21 | 54 | DM47 | 8 | DD63 | 6 |
| DC53 | 68 | DD21 | 62 | DC79 | 22 | DC83 | 14 | DD63 | 54 | BW19 | 8 | DD67 | 6 |
| E102 | 69 | CA25 | 62 | DD75 | 22 | AY70 | 14 | DD62 | 54 | DC65 | 8 | DC72 | 6 |
| DC65 | 69 | BY46 | 62 | DC53 | 22 | DM45 | 14 | DC49 | 54 | DC51 | 8 | DD41 | 6 |
| E104 | 69 | DD04 | 63 | DD44 | 22 | DC75 | 14 | DC85 | 54 | DM85 | 8 | CA25 | 6 |
| DD61 | 70 | DC96 | 63 | DM77 | 23 | DC51 | 14 | DC95 | 54 | DD79 | 8 | DC48 | 6 |
| BY46 | 70 | DC39 | 63 | CE62 | 23 | DJ89 | 14 | DC71 | 55 | DC63 | 8 | DD05 | 6 |
| DC97 | 71 | DC14 | 63 | DM67 | 23 | DC71 | 14 | DD82 | 55 | DD47 | 8 | DC43 | 6 |
| DC93 | 71 | DC84 | 63 | DC63 | 23 | DD46 | 14 | DC39 | 55 | AY7C | 8 | DC38 | 6 |
| DD66 | 72 | DD40 | 65 | DD81 | 23 | DD32 | 14 | DJ87 | 56 | DC84 | 8 | DC09 | 6 |
| DD13 | 73 | L693 | 65 | DD68 | 23 | DD43 | 14 | DC75 | 57 | DC42 | 8 | DJ48 | 6 |
| DM77 | 73 | DD78 | 65 | C145 | 23 | DD18 | 15 | DD05 | 57 | DD51 | 8 | DJ74 | 6 |

| TOTAL OPERS | | TOTAL OPNDS | | UNIQ OPERS | | UNIQ OPNDS | | CODE LINES | | DECL VARS | | FOR STMTS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DC39 | 75 | E104 | 66 | DC95 | 23 | DC96 | 15 | E104 | 57 | DC83 | 8 | DD46 | 6 |
| DC43 | 75 | DD07 | 66 | CS96 | 23 | C102 | 15 | DJ75 | 58 | DC64 | 8 | DC53 | 6 |
| CC20 | 76 | DD66 | 66 | DD51 | 23 | DC55 | 15 | DC14 | 58 | DJ45 | 8 | DC14 | 6 |
| DD68 | 76 | DC65 | 67 | DD13 | 23 | DD11 | 15 | DN02 | 58 | DC81 | 8 | DC92 | 6 |
| DD11 | 76 | DD23 | 67 | DC14 | 23 | DD47 | 15 | DD41 | 58 | DD83 | 9 | DC91 | 6 |
| DD48 | 76 | Y265 | 67 | DC49 | 23 | DC84 | 15 | DA47 | 58 | DJ89 | 9 | DC66 | 6 |
| C145 | 77 | DP06 | 68 | DJ74 | 23 | DH02 | 15 | DD01 | 60 | C102 | 9 | CS95 | 6 |
| DJ75 | 77 | DC43 | 68 | DD04 | 23 | DM77 | 15 | DD52 | 60 | DD32 | 9 | DK22 | 7 |
| Y265 | 78 | DM77 | 68 | DC96 | 23 | DC92 | 15 | DJ94 | 60 | AR28 | 9 | BY46 | 7 |
| L693 | 78 | DD32 | 68 | DD82 | 23 | E102 | 15 | DD78 | 61 | CM77 | 9 | DC05 | 7 |
| DD78 | 78 | CS96 | 70 | DD07 | 23 | DD27 | 15 | DC96 | 61 | DD76 | 9 | DD26 | 7 |
| DD04 | 78 | DC41 | 70 | BP72 | 23 | AR28 | 15 | DP28 | 61 | DP26 | 9 | DJ94 | 7 |
| DC52 | 79 | DD13 | 70 | DD87 | 23 | DD11 | 15 | Y265 | 62 | DD11 | 9 | DC79 | 7 |
| DC41 | 79 | CC20 | 71 | DD83 | 23 | DD77 | 15 | DD33 | 62 | CE62 | 9 | CE62 | 7 |
| DD64 | 79 | DD68 | 71 | DM47 | 23 | CE62 | 15 | DP06 | 62 | DJ87 | 9 | DP72 | 7 |
| DC95 | 80 | DJ75 | 71 | DC42 | 23 | DD13 | 15 | DC45 | 62 | BY46 | 9 | DC81 | 7 |
| DC96 | 81 | DD03 | 72 | DD36 | 24 | DD03 | 15 | DM77 | 62 | DD78 | 9 | DD82 | 7 |
| DC14 | 81 | C145 | 72 | DD11 | 24 | DA47 | 16 | DD66 | 62 | DC96 | 9 | DD43 | 7 |
| DP08 | 81 | DD16 | 73 | DC86 | 24 | DD21 | 16 | DC97 | 63 | DJ94 | 9 | DD22 | 7 |
| DC55 | 82 | DJ94 | 74 | Z926 | 24 | DD26 | 16 | DD48 | 64 | DD53 | 10 | DD18 | 7 |
| CS98 | 82 | DN02 | 74 | DC68 | 24 | DD39 | 16 | CS96 | 65 | DD39 | 10 | DC39 | 7 |
| DJ94 | 83 | DC95 | 74 | DC39 | 24 | DD12 | 16 | DC93 | 66 | DD26 | 10 | DP26 | 7 |
| DD32 | 84 | DD64 | 75 | L693 | 24 | DC27 | 16 | E102 | 66 | DD06 | 10 | DC82 | 7 |
| DD26 | 86 | AR28 | 76 | DD77 | 24 | DD36 | 16 | DD18 | 67 | DC85 | 10 | L693 | 7 |
| DN02 | 86 | DC86 | 76 | DD41 | 24 | DD49 | 16 | DC99 | 69 | DD48 | 10 | DD32 | 7 |
| DD18 | 87 | DD11 | 78 | DC66 | 24 | BY48 | 16 | L693 | 69 | DA47 | 10 | DC86 | 7 |
| DD43 | 88 | DD18 | 79 | DD56 | 24 | Z926 | 16 | AR28 | 70 | DD33 | 10 | DC95 | 8 |
| DC89 | 89 | DD06 | 81 | DD23 | 24 | DD53 | 17 | DD43 | 72 | DC93 | 10 | DDC7 | 8 |
| AR28 | 90 | DD83 | 81 | DJ75 | 24 | DD05 | 17 | DD06 | 73 | DD68 | 10 | DJ75 | 8 |
| DD08 | 92 | DJ89 | 81 | DJ94 | 24 | DD33 | 17 | DC86 | 73 | DC79 | 10 | DD83 | 8 |
| DJ89 | 93 | DD26 | 84 | DD64 | 25 | E104 | 17 | DD32 | 75 | DD62 | 11 | DJ89 | 8 |
| DP28 | 93 | DC55 | 88 | DD16 | 25 | DD62 | 17 | DC52 | 78 | E103 | 11 | DD78 | 8 |
| DD18 | 98 | DD43 | 88 | AR28 | 25 | E103 | 18 | DD36 | 88 | DC97 | 11 | DD52 | 9 |
| DD83 | 107 | DP26 | 92 | CC20 | 25 | DJ94 | 18 | DC82 | 89 | DN02 | 11 | DD16 | 9 |
| DC82 | 112 | DC82 | 111 | DC43 | 25 | DC93 | 18 | DJ89 | 90 | DD12 | 11 | DC52 | 9 |
| DD38 | 121 | Z926 | 118 | DC52 | 26 | DD92 | 19 | Z926 | 90 | E104 | 11 | AR28 | 9 |
| Z928 | 123 | DD36 | 121 | DD26 | 28 | DC67 | 19 | DD83 | 97 | DD82 | 14 | DD38 | 12 |

WARNING: HEADING(S) ARE NOT CORRECT. AT LEAST ONE REFLECTS A SUM OF TWO COUNTS.




FREQUENCY DISTRIBUTION GRAPH FOR PAIRS OF PROGRAMS

```
THE FOLLOWING PAIRS HAVE A CORRELATION OF 28:
        Z200.Z201
        Z200.Z202
        Z203.Z204
        Z205.Z206

THE FOLLOWING PAIRS HAVE A CORRELATION OF 29:
        Z203.Z207

THE FOLLOWING PAIRS HAVE A CORRELATION OF 30:
        Z201.Z202

        NOTE:  IDENTIFIER NAMES HAVE BEEN CHANGED
```

# APPENDIX D

## THIRD LISTING

| | TOTAL LINES | CODE LINES | CODE COMNT LINES | MULT STMT LINES | CONS AND TYPES | DECL VARS | VARS NOT USED | PROCS AND FUNCS | VAR PARAM | VAL PARAM | PROC VARS | FOR STMTS | REP STMTS | WHILE STMTS | GOTO STMTS | UNIQ OPERS | UNIQ OPNDS | TOTAL OPERS | TOTAL OPNDS | INVENTING FUNCTION |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1429 | 222 | 52 | 54 | 0 | 2 | 30 | 0 | 4 | 6 | 8 | 18 | 5 | 0 | 5 | 0 | 25 | 34 | 88 | 74 | 69 |
| 1426 | 176 | 55 | 65 | 0 | 5 | 27 | 0 | 4 | 6 | 8 | 17 | 5 | 0 | 5 | 0 | 24 | 29 | 88 | 73 | 73 |
| 1421 | 137 | 57 | 35 | 2 | 5 | 28 | 2 | 4 | 8 | 6 | 16 | 5 | 0 | 5 | 0 | 26 | 30 | 89 | 73 | 74 * |
| 1430 | 123 | 57 | 35 | 3 | 2 | 27 | 1 | 4 | 6 | 8 | 17 | 5 | 0 | 5 | 0 | 25 | 31 | 90 | 77 | 74 * |
| 1433 | 155 | 58 | 52 | 2 | 5 | 28 | 1 | 4 | 6 | 8 | 18 | 4 | 0 | 5 | 0 | 25 | 30 | 83 | 72 | 74 * |
| 1417 | 113 | 57 | 24 | 0 | 4 | 29 | 1 | 4 | 6 | 8 | 17 | 5 | 0 | 5 | 0 | 24 | 31 | 87 | 72 | 75 * |
| 1440 | 162 | 58 | 40 | 0 | 3 | 27 | 1 | 4 | 6 | 8 | 17 | 5 | 0 | 5 | 0 | 25 | 31 | 85 | 71 | 75 * |
| 1402 | 145 | 59 | 30 | 0 | 4 | 28 | 1 | 4 | 6 | 8 | 18 | 5 | 0 | 5 | 0 | 24 | 30 | 87 | 72 | 77 * |
| 1422 | 184 | 72 | 42 | 0 | 2 | 23 | 3 | 4 | 6 | 4 | 13 | 5 | 0 | 5 | 0 | 26 | 27 | 86 | 71 | 84 * |
| 1410 | 126 | 57 | 44 | 0 | 3 | 26 | 0 | 4 | 6 | 8 | 16 | 5 | 0 | 5 | 0 | 25 | 30 | 90 | 77 | [illegible] |
| 1425 | 146 | 57 | 53 | 0 | 3 | 27 | 0 | 4 | 6 | 8 | 17 | 5 | 0 | 5 | 0 | 25 | 31 | 87 | 73 | [illegible] |
| 1411 | 267 | 54 | 60 | 0 | 2 | 30 | 0 | 4 | 6 | 9 | 20 | 7 | 0 | 4 | 0 | 24 | 35 | 89 | 74 | [illegible] |
| 1418 | 130 | 57 | 40 | 0 | 3 | 31 | 0 | 4 | 6 | 8 | 21 | 7 | 0 | 4 | 0 | 24 | 35 | 88 | 71 | [illegible] |
| 1413 | 182 | 64 | 65 | 0 | 2 | 37 | 0 | 5 | 6 | 14 | 20 | 5 | 0 | 5 | 0 | 26 | 40 | 97 | 80 | [illegible] |
| 1420 | 155 | 57 | 36 | 0 | 4 | 27 | 0 | 4 | 6 | 8 | 17 | 5 | 0 | 5 | 0 | 25 | 31 | 85 | 73 | [illegible] |
| 1437 | 157 | 59 | 78 | 1 | 5 | 28 | 0 | 4 | 6 | 8 | 17 | 5 | 1 | 4 | 0 | 28 | 32 | 95 | 81 | [illegible] |
| 1432 | 165 | 56 | 49 | 0 | 2 | 24 | 4 | 4 | 13 | 1 | 14 | 5 | 0 | 5 | 0 | 27 | 28 | 96 | 82 | [illegible] * |
| 1409 | 122 | 66 | 27 | 0 | 4 | 31 | 0 | 4 | 7 | 8 | 21 | 5 | 0 | 5 | 0 | 24 | 33 | 89 | 73 | [illegible] |
| 1415 | 187 | 59 | 57 | 0 | 2 | 28 | 0 | 4 | 6 | 8 | 18 | 5 | 0 | 5 | 0 | 24 | 32 | 86 | 72 | [illegible] |
| 1428 | 159 | 58 | 40 | 0 | 3 | 31 | 1 | 4 | 6 | 8 | 19 | 5 | 0 | 5 | 0 | 25 | 34 | 93 | 74 | [illegible] * |
| 1416 | 108 | 56 | 22 | 1 | 2 | 28 | 2 | 4 | 6 | 8 | 18 | 6 | 0 | 3 | 0 | 24 | 32 | 83 | 71 | [illegible] * |
| 1401 | 145 | 67 | 37 | 1 | 2 | 30 | 0 | 4 | 7 | 7 | 17 | 7 | 0 | 6 | 0 | 25 | 34 | 118 | 102 | [illegible] |
| 1435 | 99 | 56 | 19 | 0 | 2 | 30 | 0 | 4 | 6 | 8 | 20 | 5 | 0 | 5 | 0 | 25 | 34 | 100 | 89 | [illegible] |
| 14[illegible]8 | 113 | 53 | 25 | 0 | 3 | 27 | 0 | 4 | 6 | 8 | 17 | 5 | 0 | 4 | 0 | 24 | 31 | 86 | 71 | [illegible] |
| 1406 | 158 | 58 | 21 | 0 | 5 | 27 | 1 | 4 | 6 | 8 | 17 | 5 | 0 | 5 | 0 | 24 | 29 | 87 | 72 | 17012049 * |
| 1407 | 131 | 58 | 37 | 0 | 2 | 23 | 4 | 4 | 6 | 5 | 12 | 5 | 1 | 3 | 0 | 25 | 27 | 86 | 70 | [illegible] * |
| 1436 | 177 | 64 | 25 | 0 | 2 | 28 | 1 | 4 | 6 | 8 | 16 | 5 | 0 | 5 | 0 | 24 | 30 | 84 | 69 | [illegible] * |
| 14[illegible]8 | 106 | 49 | 23 | 2 | 2 | 28 | 0 | 4 | 6 | 8 | 18 | 5 | 1 | 4 | 0 | 27 | 32 | 86 | 73 | [illegible] |
| 1441 | 145 | 55 | 36 | 0 | 5 | 26 | 2 | 4 | 8 | 5 | 16 | 5 | 0 | 5 | 0 | 24 | 28 | 88 | 72 | [illegible] * |
| 1419 | 162 | 55 | 55 | 0 | 5 | 29 | 0 | 4 | 6 | 8 | 19 | 5 | 1 | 4 | 0 | 27 | 31 | 88 | 75 | [illegible] |
| 1[illegible]12 | 165 | 65 | 29 | 0 | 3 | 27 | 2 | 4 | 6 | 8 | 17 | 5 | 1 | 4 | 0 | 28 | 31 | 102 | 88 | [illegible] * |
| 1427 | 189 | 65 | 27 | 0 | 4 | 30 | 2 | 4 | 7 | 7 | 18 | 5 | 0 | 5 | 0 | 26 | 32 | 96 | 80 | [illegible] * |
| 1408 | 154 | 56 | 24 | 0 | 2 | 27 | 0 | 4 | 6 | 8 | 17 | 5 | 1 | 4 | 0 | 27 | 32 | 92 | 84 | [illegible] |
| 1431 | 127 | 57 | 21 | 0 | 2 | 30 | 2 | 4 | 6 | 8 | 18 | 5 | 0 | 5 | 0 | 26 | 34 | 85 | 71 | [illegible] * |
| 1423 | 194 | 60 | 56 | 1 | 3 | 27 | 0 | 4 | 6 | 8 | 14 | 5 | 1 | 4 | 0 | 28 | 31 | 95 | 81 | [illegible] |
| 1404 | 146 | 56 | 48 | 1 | 5 | 27 | 3 | 4 | 6 | 8 | 17 | 5 | 0 | 5 | 0 | 26 | 31 | 85 | 72 | [illegible] * |
| 1414 | 109 | 55 | 26 | 0 | 2 | 30 | 0 | 4 | 6 | 8 | 20 | 7 | 1 | 3 | 0 | 25 | 34 | 87 | 74 | [illegible] |
| 1434 | 124 | 59 | 43 | 0 | 2 | 28 | 0 | 4 | 6 | 8 | 18 | 5 | 0 | 5 | 0 | 25 | 32 | 93 | 76 | [illegible] |
| 1439 | 174 | 58 | 58 | 0 | 7 | 26 | 1 | 4 | 6 | 8 | 16 | 4 | 0 | 5 | 0 | 25 | 30 | 90 | 82 | [illegible] * |
| 1424 | 187 | 60 | 80 | 0 | 3 | 28 | 0 | 4 | 6 | 8 | 17 | 5 | 2 | 3 | 0 | 26 | 33 | 97 | 85 | [illegible] |
| 1405 | 189 | 56 | 41 | 0 | 5 | 29 | 0 | 4 | 6 | 8 | 19 | 5 | 1 | 4 | 0 | 26 | 31 | 90 | 78 | 22029027 |
| 1412 | 291 | 59 | 80 | 0 | 3 | 30 | 1 | 4 | 6 | 8 | 20 | 7 | 1 | 2 | 0 | 27 | 35 | 91 | 74 | [illegible] * |
| 1403 | 143 | 61 | 32 | 0 | 2 | 29 | 0 | 4 | 6 | 8 | 19 | 6 | 0 | 5 | 0 | 26 | 35 | 102 | 82 | [illegible] |

* VARIABLE(S) WERE DECLARED BUT NOT USED

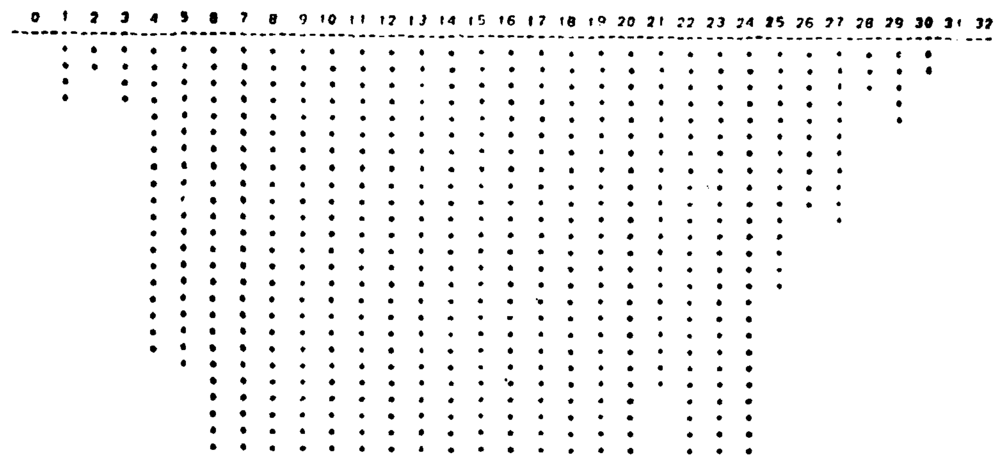
** PROCEDURE(S) WERE DECLARED BUT NOT USED

*** VARIABLE(S) AND PROCEDURE(S) WERE DECLARED BUT NOT USED

| TOTAL OPERS | | TOTAL OPNDS | | UNIQ OPERS | | UNIQ OPNDS | | CODE LINES | | DECL VARS | | FOR STMTS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1416 | 83 | 1436 | 69 | 1406 | 24 | 1407 | 27 | 1438 | 49 | 1407 | 23 | 1439 | 9 |
| 1418 | 83 | 1407 | 70 | 1415 | 24 | 1422 | 27 | 1429 | 52 | 1422 | 23 | 1416 | 9 |
| 1436 | 84 | 1422 | 71 | 1402 | 24 | 1411 | 28 | 1428 | 53 | 1432 | 24 | 1428 | 9 |
| 1440 | 85 | 1431 | 71 | 1417 | 24 | 1432 | 28 | 1411 | 54 | 1439 | 26 | 1407 | 9 |
| 1420 | 85 | 1428 | 71 | 1409 | 24 | 1426 | 29 | 1414 | 55 | 1421 | 26 | 1433 | 9 |
| 1404 | 85 | 1416 | 71 | 1411 | 24 | 1406 | 29 | 1441 | 55 | 1410 | 26 | 1415 | 10 |
| 1431 | 85 | 1440 | 71 | 1441 | 24 | 1410 | 30 | 1419 | 55 | 1436 | 26 | 1405 | 10 |
| 1428 | 86 | 1418 | 71 | 1426 | 24 | 1439 | 30 | 1426 | 55 | 1441 | 26 | 1424 | 10 |
| 1415 | 86 | 1402 | 72 | 1428 | 24 | 1421 | 30 | 1405 | 56 | 1440 | 27 | 1438 | 10 |
| 1429 | 86 | 1417 | 72 | 1436 | 24 | 1433 | 30 | 1416 | 56 | 1430 | 27 | 1437 | 10 |
| 1441 | 86 | 1406 | 72 | 1416 | 24 | 1436 | 30 | 1404 | 56 | 1425 | 27 | 1402 | 10 |
| 1419 | 86 | 1415 | 72 | 1418 | 24 | 1402 | 30 | 1432 | 56 | 1420 | 27 | 1419 | 10 |
| 1407 | 86 | 1404 | 72 | 1429 | 25 | 1440 | 31 | 1435 | 56 | 1406 | 27 | 1409 | 10 |
| 1438 | 86 | 1433 | 72 | 1E28 | 25 | 1423 | 31 | 1408 | 56 | 1408 | 27 | 1412 | 10 |
| 1422 | 86 | 1441 | 72 | 1425 | 25 | 1428 | 31 | 1430 | 57 | 1426 | 27 | 1413 | 10 |
| 1406 | 87 | 1426 | 73 | 1435 | 25 | 1419 | 31 | 1418 | 57 | 1442 | 27 | 1E28 | 10 |
| 1425 | 87 | 1409 | 73 | 1434 | 25 | 1417 | 31 | 1425 | 57 | 1428 | 27 | 1431 | 10 |
| 1402 | 87 | 1421 | 73 | 1410 | 25 | 1420 | 31 | 1431 | 57 | 1404 | 27 | 1429 | 10 |
| 1414 | 87 | 1438 | 73 | 1414 | 25 | 1405 | 31 | 1410 | 57 | 1423 | 27 | 1417 | 10 |
| 1417 | 87 | 1420 | 73 | 1430 | 25 | 1430 | 31 | 1421 | 57 | 1433 | 28 | 1435 | 10 |
| 1418 | 88 | 1425 | 73 | 1439 | 25 | 1442 | 31 | 1420 | 57 | 1434 | 28 | 1427 | 10 |
| 1426 | 88 | 1412 | 74 | 1433 | 25 | 1425 | 31 | 1417 | 57 | 1437 | 28 | 1432 | 10 |
| 1411 | 89 | 1E28 | 74 | 1440 | 25 | 1404 | 31 | 1439 | 58 | 1438 | 28 | 1421 | 10 |
| 1421 | 89 | 1429 | 74 | 1401 | 25 | 1415 | 32 | 1433 | 58 | 1416 | 28 | 1425 | 10 |
| 1409 | 89 | 1414 | 74 | 1420 | 25 | 1416 | 32 | 1E28 | 58 | 1402 | 28 | 1430 | 10 |
| 1405 | 90 | 1411 | 74 | 1427 | 26 | 1438 | 32 | 1407 | 58 | 1415 | 28 | 1436 | 10 |
| 1439 | 90 | 1419 | 75 | 1403 | 26 | 1437 | 32 | 1406 | 58 | 1424 | 28 | 1410 | 10 |
| 1430 | 90 | 1434 | 76 | 1405 | 26 | 1409 | 32 | 1440 | 58 | 1403 | 29 | 1440 | 10 |
| 1410 | 90 | 1410 | 77 | 1413 | 26 | 1434 | 32 | 1412 | 59 | 1405 | 29 | 1441 | 10 |
| 1412 | 91 | 1430 | 77 | 1431 | 26 | 1427 | 32 | 1437 | 59 | 1419 | 29 | 1420 | 10 |
| 1408 | 92 | 1405 | 78 | 1422 | 26 | 1408 | 33 | 1415 | 59 | 1417 | 29 | 1404 | 10 |
| 1E28 | 93 | 1427 | 80 | 1407 | 26 | 1424 | 33 | 1402 | 59 | 1414 | 30 | 1442 | 10 |
| 1434 | 93 | 1413 | 80 | 1421 | 26 | 1435 | 34 | 1434 | 59 | 1429 | 30 | 1434 | 10 |
| 1437 | 95 | 1423 | 81 | 1424 | 26 | 1E28 | 34 | 1423 | 60 | 1411 | 30 | 1423 | 10 |
| 1423 | 95 | 1437 | 81 | 1404 | 26 | 1429 | 34 | 1424 | 60 | 1401 | 30 | 1426 | 10 |
| 1427 | 96 | 1403 | 82 | 1408 | 27 | 1431 | 34 | 1403 | 61 | 1427 | 30 | 1406 | 10 |
| 1432 | 96 | 1432 | 82 | 1432 | 27 | 1401 | 34 | 1413 | 64 | 1435 | 30 | 1422 | 10 |
| 1424 | 97 | 1439 | 82 | 1412 | 27 | 1414 | 34 | 1436 | 64 | 1431 | 30 | 1408 | 10 |
| 1413 | 97 | 1408 | 84 | 1438 | 27 | 1411 | 35 | 1442 | 65 | 1412 | 30 | 1418 | 11 |
| 1435 | 100 | 1424 | 85 | 1419 | 27 | 1418 | 35 | 1427 | 65 | 1409 | 31 | 1414 | 11 |
| 1403 | 102 | 1442 | 88 | 1437 | 28 | 1403 | 35 | 1409 | 66 | 1E28 | 31 | 1403 | 11 |
| 1442 | 102 | 1435 | 89 | 1423 | 28 | 1412 | 35 | 1401 | 67 | 1418 | 31 | 1411 | 11 |
| 1401 | 116 | 1401 | 102 | 1442 | 28 | 1413 | 40 | 1422 | 72 | 1413 | 37 | 1401 | 13 |

WARNING: HEADING(S) ARE NOT CORRECT. AT LEAST ONE REFLECTS A SUM OF TWO COUNTS.

FREQUENCY DISTRIBUTION GRAPH FOR PAIRS OF PROGRAMS



```
THE FOLLOWING PAIRS HAVE A CORRELATION OF 28:
          T402,T417
          T404,T440
          T417,T425

THE FOLLOWING PAIRS HAVE A CORRELATION OF 29:
          T402,T408
          T402,T415
          T404,T420
          T420,T440
          T423,T437

THE FOLLOWING PAIRS HAVE A CORRELATION OF 30:
          T410,T430
          T420,T425
```

# APPENDIX E

## FOURTH LISTING

| | TOTAL LINES | CODE LINES | CODE COMNT LINES | MULT STMT LINES | CONS AND TYPES | DECL VARS | VARS NOT USED | PROCS AND FUNCS | VAR PARAM | VAL PARAM | PROC VARS | FOR STMTS | REP STMTS | WHILE STMTS | GOTO STMTS | UNIO OPERS | UNIO OPNDS | TOTAL OPERS | TOTAL OPNDS | INDENTING FUNCTION |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| H130 | 167 | 51 | 63 | 0 | 3 | 11 | 0 | 2 | 3 | 2 | 8 | 0 | 0 | 4 | 0 | 25 | 13 | 113 | 95 | [illegible] |
| H107 | 133 | 44 | 38 | 0 | 4 | 17 | 2 | 3 | 9 | 1 | 8 | 1 | 0 | 3 | 0 | 25 | 21 | 79 | 84 | 2010038 * |
| H129 | 174 | 68 | 39 | 0 | 4 | 23 | 1 | 4 | 7 | 4 | 16 | 0 | 0 | 6 | 0 | 26 | 26 | 115 | 80 | [illegible]29069 * |
| H122 | 235 | 80 | 84 | 0 | 4 | 33 | 0 | 5 | 11 | 6 | 19 | 3 | 0 | 3 | 0 | 30 | 36 | 147 | 110 | 12[illegible]1077 |
| H114 | 175 | 51 | 49 | 0 | 4 | 19 | 3 | 5 | 3 | 5 | 13 | 3 | 0 | 2 | 0 | 28 | 23 | 67 | 48 | 13012041 * |
| H110 | 158 | 57 | 39 | 0 | 4 | 20 | 1 | 5 | 7 | 4 | 12 | 4 | 1 | 1 | 0 | 30 | 25 | 96 | 71 | 14030054 * |
| H128 | 131 | 53 | 32 | 0 | 3 | 15 | 2 | 4 | 1 | 4 | 8 | 3 | 0 | 3 | 0 | 26 | 18 | 64 | 50 | 14[illegible]40 * |
| H124 | 162 | 60 | 45 | 0 | 4 | 20 | 2 | 4 | 5 | 4 | 11 | 3 | 0 | 3 | 0 | 28 | 23 | 112 | 82 | 15016045 * |
| H103 | 158 | 57 | 41 | 0 | 4 | 25 | 0 | 3 | 6 | 2 | 14 | 4 | 0 | 3 | 0 | 29 | 28 | 117 | 94 | 16011049 |
| H109 | 138 | 60 | 33 | 0 | 3 | 13 | 0 | 2 | 2 | 0 | 4 | 8 | 0 | 5 | 0 | 25 | 16 | 127 | 78 | [illegible] |
| H115 | 163 | 60 | 48 | 0 | 4 | 18 | 0 | 3 | 4 | 0 | 7 | 3 | 0 | 2 | 0 | 26 | 21 | 93 | 70 | [illegible] |
| H119 | 179 | 70 | 43 | 0 | 4 | 13 | 6 | 4 | 6 | 2 | 5 | 3 | 0 | 3 | 0 | 28 | 16 | 95 | 63 | [illegible] * |
| H118 | 182 | 53 | 55 | 0 | 3 | 25 | 1 | 4 | 8 | 5 | 17 | 3 | 0 | 5 | 0 | 26 | 27 | 105 | 65 | [illegible] * |
| H132 | 124 | 44 | 36 | 0 | 3 | 7 | 0 | 5 | 1 | 3 | 4 | 0 | 0 | 2 | 0 | 30 | 10 | 96 | 58 | [illegible]24 |
| H102 | 174 | 62 | 43 | 0 | 3 | 22 | 1 | 3 | 5 | 2 | 10 | 3 | 0 | 3 | 0 | 27 | 27 | 95 | 72 | 19[illegible]054 * |
| H106 | 148 | 51 | 27 | 1 | 3 | 20 | 0 | 3 | 5 | 3 | 10 | 0 | 0 | 4 | 0 | 26 | 23 | 101 | 74 | [illegible]031 |
| H127 | 136 | 68 | 37 | 0 | 3 | 8 | 6 | 3 | 2 | 0 | 3 | 4 | 0 | 3 | 0 | 27 | 12 | 103 | 66 | [illegible]51 * |
| H105 | 172 | 64 | 56 | 0 | 2 | 18 | 0 | 2 | 3 | 0 | 7 | 1 | 1 | 3 | 3 | 27 | 20 | 90 | 69 | [illegible]12 |
| H123 | 204 | 48 | 56 | 0 | 4 | 17 | 6 | 5 | 8 | 5 | 12 | 6 | 0 | 3 | 0 | 27 | 21 | 71 | 49 | [illegible] * |
| H133 | 187 | 62 | 62 | 0 | 5 | 24 | 1 | 5 | 5 | 7 | 16 | 0 | 0 | 4 | 0 | 30 | 27 | 109 | 82 | [illegible] * |
| H116 | 167 | 66 | 42 | 0 | 4 | 15 | 2 | 5 | 4 | 2 | 8 | 0 | 0 | 5 | 0 | 27 | 18 | 87 | 68 | [illegible]040 * |
| H101 | 179 | 64 | 39 | 0 | 5 | 24 | 3 | 4 | 11 | 2 | 15 | 5 | 0 | 1 | 0 | 28 | 29 | 133 | 86 | 21015045 * |
| H120 | 201 | 69 | 58 | 0 | 4 | 18 | 0 | 4 | 3 | 3 | 11 | 1 | 0 | 4 | 0 | 29 | 22 | 107 | 71 | 2101[illegible]050 |
| H125 | 178 | 60 | 47 | 0 | 3 | 26 | 0 | 4 | 11 | 1 | 16 | 4 | 0 | 2 | 0 | 28 | 28 | 92 | 63 | 21020037 |
| H113 | 238 | 88 | 44 | 0 | 4 | 18 | 2 | 4 | 2 | 2 | 8 | 0 | 0 | 5 | 0 | 31 | 21 | 151 | 102 | 21021068 * |
| H111 | 146 | 62 | 48 | 0 | 3 | 12 | 4 | 4 | 4 | 2 | 6 | 0 | 0 | 5 | 0 | 26 | 15 | 108 | 75 | [illegible]37 * |
| H112 | 178 | 71 | 37 | 0 | 4 | 15 | 2 | 4 | 4 | 4 | 8 | 3 | 0 | 4 | 0 | 30 | 20 | 131 | 94 | [illegible] * |
| H1[illegible] | 133 | 68 | 34 | 0 | 3 | 10 | 0 | 1 | 2 | 0 | 2 | 0 | 0 | 6 | 0 | 22 | 15 | 114 | 83 | [illegible]23036 |
| H1[illegible] | 15[illegible] | 70 | 41 | 0 | 3 | 20 | 3 | 5 | 6 | 3 | 12 | 2 | 0 | 6 | 0 | 28 | 22 | 102 | 71 | [illegible]22041 * |
| H1[illegible] | [illegible] | 82 | 55 | 0 | 3 | 20 | 1 | 4 | 3 | 6 | 10 | 4 | 0 | 2 | 0 | 29 | 23 | 121 | 88 | [illegible]24050 * |
| H1[illegible] | [illegible] | 73 | 51 | 0 | 4 | 22 | 4 | 5 | 9 | 0 | 14 | 3 | 0 | 3 | 0 | 28 | 25 | 108 | 71 | [illegible]25043 * |
| H1 8 | [illegible] | 76 | 55 | 0 | 3 | 23 | 0 | 4 | 3 | 4 | 14 | 3 | 0 | 4 | 0 | 25 | 25 | 97 | 68 | [illegible]045 |
| H1[illegible] | [illegible] | 95 | 53 | 0 | 8 | 32 | 5 | 6 | 13 | 8 | 23 | 4 | 0 | 3 | 0 | 31 | 35 | 199 | 168 | [illegible]54 * |

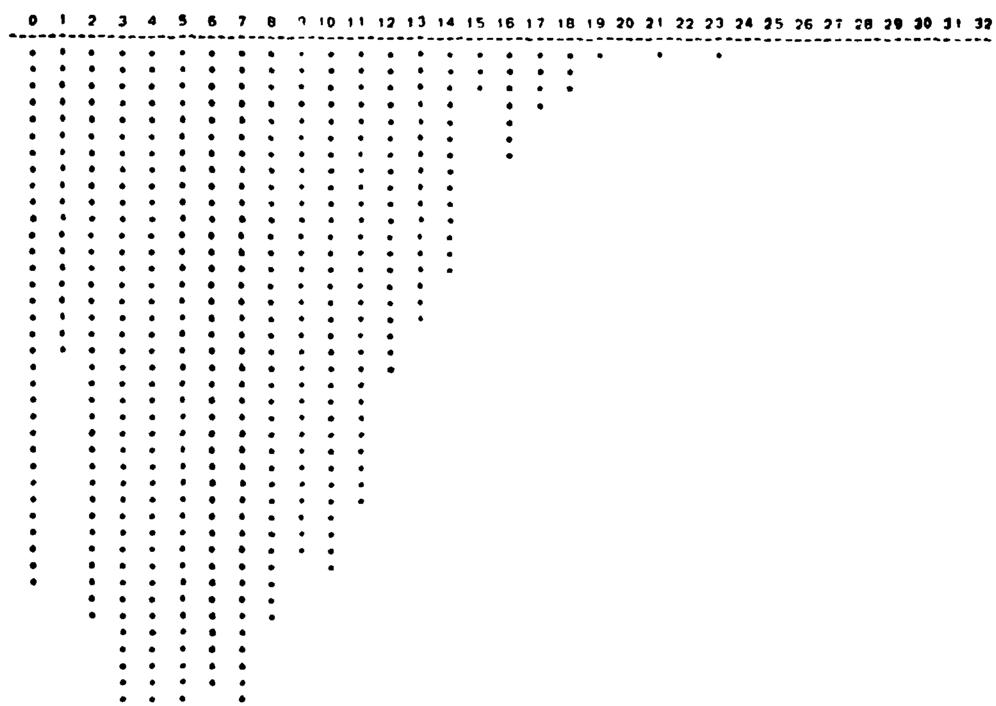
* VARIABLE(S) WERE DECLARED BUT NOT USED
** PROCEDURE(S) WERE DECLARED BUT NOT USED
*** VARIABLE(S) AND PROCEDURE(S) WERE DECLARED BUT NOT USED

| | TOTAL OPERS | | TOTAL OPNDS | | UNIQ OPERS | | UNIQ OPNDS | | CODE LINES | | DECL VARS | | FOR STMTS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| H126 | 64 | H114 | 48 | H117 | 22 | H132 | 10 | H107 | 44 | H132 | 7 | H132 | 2 |
| H114 | 67 | H123 | 49 | H107 | 25 | H127 | 12 | H132 | 44 | H127 | 8 | H130 | 4 |
| H123 | 71 | H126 | 50 | H130 | 25 | H130 | 13 | H123 | 49 | H117 | 10 | H107 | 4 |
| H107 | 79 | H132 | 58 | H115 | 26 | H117 | 15 | H114 | 51 | H130 | 11 | H133 | 4 |
| H116 | 87 | H125 | 63 | H129 | 26 | H111 | 15 | H106 | 51 | H111 | 12 | H106 | 4 |
| H105 | 90 | H107 | 64 | H106 | 26 | H109 | 16 | H130 | 51 | H109 | 13 | H113 | 5 |
| H125 | 92 | H118 | 68 | H111 | 26 | H119 | 16 | H126 | 53 | H119 | 13 | H105 | 5 |
| H115 | 93 | H127 | 68 | H109 | 26 | H126 | 18 | H118 | 53 | H116 | 15 | H114 | 5 |
| H119 | 95 | H128 | 68 | H128 | 26 | H116 | 18 | H103 | 57 | H112 | 15 | H115 | 5 |
| H102 | 95 | H116 | 68 | H118 | 26 | H105 | 20 | H110 | 57 | H126 | 15 | H120 | 5 |
| H132 | 96 | H105 | 69 | H126 | 26 | H112 | 20 | H125 | 60 | H105 | 16 | H116 | 5 |
| H110 | 96 | H119 | 69 | H123 | 27 | H115 | 21 | H124 | 60 | H113 | 16 | H111 | 5 |
| H128 | 97 | H115 | 70 | H116 | 27 | H107 | 21 | H109 | 60 | H123 | 17 | H104 | 6 |
| H106 | 101 | H120 | 71 | H105 | 27 | H123 | 21 | H115 | 60 | H107 | 17 | H110 | 6 |
| H108 | 102 | H104 | 71 | H102 | 27 | H113 | 21 | H133 | 62 | H120 | 18 | H124 | 6 |
| H127 | 103 | H110 | 71 | H127 | 27 | H108 | 22 | H111 | 62 | H115 | 18 | H102 | 6 |
| H133 | 105 | H108 | 71 | H124 | 28 | H120 | 22 | H102 | 62 | H114 | 19 | H125 | 6 |
| H118 | 105 | H102 | 72 | H101 | 28 | H114 | 23 | H101 | 64 | H108 | 20 | H122 | 6 |
| H120 | 107 | H106 | 74 | H119 | 28 | H131 | 23 | H105 | 64 | H131 | 20 | H129 | 6 |
| H104 | 108 | H111 | 75 | H114 | 28 | H124 | 23 | H116 | 66 | H106 | 20 | H101 | 6 |
| H111 | 108 | H109 | 78 | H125 | 28 | H106 | 23 | H129 | 68 | H110 | 20 | H126 | 6 |
| H124 | 112 | H129 | 80 | H108 | 28 | H110 | 25 | H127 | 68 | H124 | 20 | H119 | 6 |
| H130 | 113 | H124 | 82 | H104 | 28 | H128 | 25 | H117 | 68 | H104 | 22 | H117 | 6 |
| H117 | 114 | H133 | 82 | H103 | 29 | H104 | 25 | H120 | 69 | H102 | 22 | H131 | 6 |
| H129 | 115 | H117 | 83 | H120 | 29 | H129 | 26 | H108 | 70 | H128 | 23 | H103 | 7 |
| H103 | 117 | H101 | 86 | H131 | 29 | H133 | 27 | H119 | 70 | H129 | 23 | H121 | 7 |
| H131 | 121 | H131 | 88 | H122 | 30 | H102 | 27 | H112 | 71 | H133 | 24 | H112 | 7 |
| H109 | 127 | H112 | 94 | H112 | 30 | H118 | 27 | H104 | 73 | H101 | 24 | H128 | 7 |
| H112 | 131 | H103 | 94 | H110 | 30 | H125 | 28 | H128 | 76 | H103 | 25 | H127 | 7 |
| H101 | 133 | H130 | 95 | H133 | 30 | H103 | 28 | H122 | 80 | H118 | 25 | H108 | 8 |
| H122 | 147 | H113 | 102 | H132 | 30 | H101 | 29 | H131 | 82 | H125 | 26 | H118 | 8 |
| H113 | 151 | H122 | 110 | H121 | 31 | H121 | 35 | H113 | 88 | H121 | 32 | H123 | 9 |
| H121 | 199 | H121 | 168 | H113 | 31 | H122 | 36 | H121 | 95 | H122 | 33 | H109 | 13 |

WARNING: HEADING(S) ARE NOT CORRECT. AT LEAST ONE REFLECTS A SUM OF TWO COUNTS.

TUKEY ESTIMATE FOR SUSPICION OF PLAGIARISM: 28

FREQUENCY DISTRIBUTION GRAPH FOR PAIRS OF PROGRAMS



# APPENDIX F

## FIFTH LISTING

| | TOTAL LINES | CODE LINES | CODE COMMT LINES | MULT STMT LINES | CONS AND TYPES | DECL VARS | VARS NOT USED | PROCS AND FUNCS | VAR PARAM | VAL PARAM | PROC VARS | FOR STMTS | REP STMTS | WHILE STMTS | GOTO STMTS | UNIQ OPERS | UNIQ OPNDS | TOTAL OPERS | TOTAL OPNDS | INDENTING FUNCTION |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| H210 | 217 | 56 | 36 | 0 | 4 | 20 | 1 | 4 | 1 | 3 | 17 | 5 | 0 | 1 | 0 | 26 | 31 | 82 | 73 | 15010070 * |
| H212 | 329 | 84 | 69 | 0 | 6 | 37 | 1 | 8 | 5 | 2 | 26 | 1 | 3 | 4 | 0 | 37 | 43 | 181 | 145 | 16013115 * |
| [illegible] | 150 | 57 | 20 | 0 | 2 | 23 | 0 | 6 | 2 | 4 | 18 | 5 | 2 | 2 | 0 | 30 | 27 | 83 | 63 | [illegible] |
| [illegible] | 200 | 63 | 50 | 0 | 4 | 27 | 0 | 8 | 4 | 3 | 23 | 4 | 0 | 4 | 0 | 38 | 35 | 139 | 114 | [illegible] |
| H211 | 204 | 75 | 79 | 0 | 4 | 39 | 1 | 6 | 6 | 3 | 31 | 6 | 0 | 3 | 0 | 33 | 45 | 152 | 136 | [illegible] * |
| [illegible] | 314 | 69 | 53 | 0 | 2 | 34 | 0 | 8 | 8 | 6 | 29 | 5 | 0 | 3 | 0 | 29 | 39 | 99 | 85 | [illegible] |
| H215 | 253 | 97 | 70 | 0 | 4 | 36 | 0 | 7 | 12 | 3 | 32 | 5 | 1 | 3 | 0 | 41 | 43 | 202 | 165 | [illegible] |
| [illegible] | 304 | 82 | 90 | 0 | 4 | 44 | 0 | 8 | 9 | 9 | 36 | 4 | 0 | 4 | 0 | 36 | 48 | 145 | 129 | [illegible] |
| [illegible] | 236 | 80 | 62 | 0 | 3 | 38 | 2 | 9 | 11 | 5 | 26 | 4 | 0 | 8 | 0 | 35 | 44 | 158 | 138 | [illegible] * |
| [illegible] | 234 | [illegible] | 55 | 0 | 3 | 20 | 0 | 8 | 5 | 3 | 19 | 3 | 2 | 2 | 0 | 34 | 33 | 166 | 145 | [illegible] |
| H217 | 312 | 66 | 84 | 0 | 5 | 32 | 1 | 8 | 8 | 4 | 27 | 2 | 1 | 5 | 0 | 37 | 36 | 165 | 126 | [illegible] * |
| [illegible] | [illegible] | 69 | 69 | 0 | 9 | 39 | 1 | 4 | 5 | 2 | 24 | 7 | 0 | 2 | 0 | 35 | 46 | 207 | 141 | [illegible] * |
| [illegible] | 356 | 124 | 0 | 0 | 7 | 39 | 0 | 10 | 9 | 8 | 23 | 9 | 1 | 2 | 0 | 41 | 44 | 295 | 279 | [illegible] |
| [illegible] | 270 | [illegible] | | 6 | 2 | 34 | 0 | 8 | 7 | 3 | 25 | 4 | 1 | 5 | 0 | 37 | 37 | 192 | 151 | [illegible] |
| [illegible] | [illegible] | 102 | [illegible] | 0 | 4 | 30 | 0 | 8 | 9 | 6 | 29 | 6 | 0 | 4 | 0 | 38 | 43 | 227 | 197 | [illegible] |
| [illegible] | [illegible] | 121 | 106 | 0 | 3 | 41 | 0 | 10 | 9 | 5 | 32 | 6 | 0 | 7 | 0 | 40 | 48 | 245 | 173 | 67047061 ** |

* VARIABLE(S) WERE DECLARED BUT NOT USED

** PROCEDURE(S) WERE DECLARED BUT NOT USED

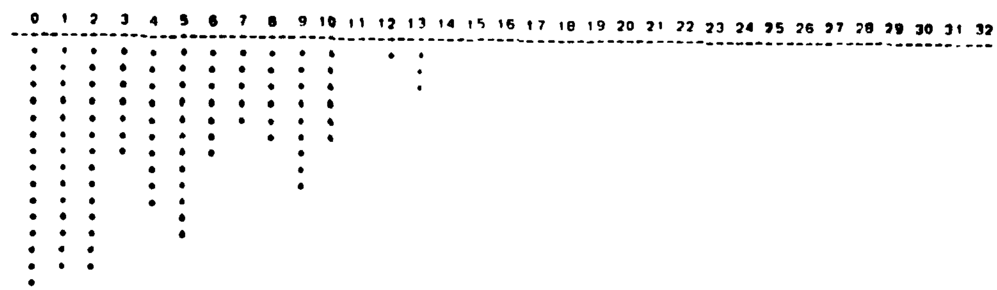
*** VARIABLE(S) AND PROCEDURE(S) WERE DECLARED BUT NOT USED

| | TOTAL OPERS | | TOTAL OPNDS | | UNIQ OPERS | | UNIQ OPNDS | | CODE LINES | | DECL VARS | | FOR STMTS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| H210 | 82 | H204 | 63 | H210 | 28 | H204 | 27 | H204 | 57 | H204 | 23 | H210 | 6 |
| H204 | 83 | H210 | 73 | H216 | 29 | H210 | 31 | H210 | 58 | H214 | 26 | H214 | 7 |
| H216 | 99 | H216 | 85 | H204 | 30 | H214 | 33 | H205 | 63 | H205 | 27 | H202 | 8 |
| H205 | 139 | H205 | 114 | H211 | 33 | H205 | 35 | H216 | 69 | H210 | 28 | H216 | 8 |
| H209 | 145 | H207 | 126 | H214 | 34 | H207 | 36 | H213 | 73 | H207 | 32 | H207 | 8 |
| H211 | 152 | H209 | 129 | H203 | 35 | H213 | 37 | H214 | 74 | H216 | 34 | H209 | 8 |
| H203 | 158 | H211 | 136 | H212 | 35 | H216 | 39 | H211 | 75 | H213 | 34 | H205 | 8 |
| H207 | 165 | H203 | 138 | H209 | 36 | H202 | 43 | H203 | 80 | H215 | 36 | H211 | 9 |
| H214 | 166 | H212 | 141 | H213 | 37 | H208 | 43 | H209 | 82 | H202 | 37 | H212 | 9 |
| H202 | 181 | H202 | 145 | H202 | 37 | H215 | 43 | H202 | 84 | H203 | 38 | H215 | 9 |
| H213 | 192 | H214 | 145 | H207 | 37 | H203 | 44 | H207 | 86 | H208 | 39 | H204 | 9 |
| H212 | 202 | H213 | 151 | H208 | 38 | H206 | 44 | H212 | 88 | H206 | 39 | H213 | 10 |
| H215 | 202 | H215 | 165 | H205 | 39 | H211 | 45 | H215 | 91 | H211 | 39 | H208 | 10 |
| H208 | 227 | H201 | 173 | H201 | 40 | H212 | 46 | H208 | 100 | H212 | 39 | H203 | 12 |
| H201 | 245 | H208 | 197 | H206 | 41 | H209 | 48 | H201 | 123 | H201 | 41 | H206 | 12 |
| H206 | 295 | H206 | 275 | H215 | 41 | H201 | 48 | H206 | 124 | H209 | 44 | H201 | 13 |

WARNING: HEADING(S) ARE NOT CORRECT.  AT LEAST ONE REFLECTS A SUM OF TWO COUNTS.

TUKEY ESTIMATE FOR SUSPICION OF PLAGIARISM: 22

FREQUENCY DISTRIBUTION GRAPH FOR PAIRS OF PROGRAMS



# APPENDIX G

## EXAMPLE OF A PAIR OF PROGRAMS WITH CORRELATION NUMBER EQUAL TO 29

```
2203

VARS

I,J
LIMIT
NEXTITEM
MAXNUMLINES
MAXNUMITEMS

BEGIN
WHILE NOT EOF DO
BEGIN
READLN(MAXNUMLINES,MAXNUMITEMS)
WRITELN("1")
IF MAXNUMLINES<0 THEN
WRITELN("INCORRECT DATA")
ELSE IF MAXNUMLINES>25 THEN
WRITELN("TO MANY LINES REQUESTED")
ELSE IF MAXNUMITEMS<0 THEN
WRITELN("INSUFFICIENT DATA")
ELSE IF MAXNUMITEMS>50 THEN
WRITELN("OVERSUFFICIENT DATA")
ELSE BEGIN
LIMIT := MAXNUMITEMS DIV 5
IF MAXNUMITEMS MOD 5 <> 0 THEN
LIMIT := LIMIT + 1;
WRITE("   0")
FOR I := 1 TO LIMIT DO
WRITE(I*5:5); WRITELN

WRITE("   :")
FOR I := 1 RO LIMIT DO
WRITE("----:")
WRITELN
WRITELN("0":3)
FOR I := 1 TO MAXNUMLINES DO BEGIN
WRITE("   ")
READ(NEXTITEM)
IF NEXTITEM<0 THEN
WRITELN("-")
ELSE BEGIN
WRITE(":")

IF NEXTITEM>MAXNUMITEMS THEN
NEXTITEM := MAXNUMITEMS
FOR J := 1 TO NEXTITEM DO
WRITE("*")
IF NEXTITEM<MAXNUMITEMS THEN
WRITE("+")
WRITELN
END
WRITELN(I:3)
END
END
READLN
END
END
```

```
2201

VARS

I,J
LIMITS
NEXTITEM
MAXNUMLNS
MAXNUMITEMS
MINITEM

BEGIN
WHILE NOT EOF DO
BEGIN
READLN(MAXNUMLNS,MAXNUMITEMS)
WRITELN("1")
IF MAXNUMLNS<0 THEN
WRITELN("INCORRECT DATA")
ELSE IF MAXNUMLNS>25 THEN
WRITELN("TOO MANY LINES REQUESTED")
ELSE IF MAXNUMITEMS<0 THEN
WRITELN("INSUFFICIENT DATA")
ELSE IF MAXNUMITEMS>50 THEN
WRITELN("OVERSUFFICIENT DATA")
ELSE BEGIN
LIMITS := MAXNUMITEMS DIV 5
IF MAXNUMITEMS MOD 5 <> 0 THEN
LIMITS := LIMITS + 1;
WRITE("   0)
FOR I := 1 TO LIMITS DO
WRITE(I*5:5)
WRITELN
WRITE("   :")
FOR I := 1 TO LIMITS DO
WRITE("----:")
WRITELN
WRITELN(" 0":3)
FOR I := 1 TO MAXNUMLNS DO BEGIN
WRITE("   ")
READ(NEXTITEM)
IF NEXTITEM<0 THEN
WRITELN("*")
ELSE BEGIN
WRITE(":")
MINITEM := NEXTITEM
IF NEXTITEM>MAXNUMITEMS THEN
MINITEM := MAXNUMITEMS
FOR J := 1 TO MINITEM DO
WRITE("*")
IF NEXTITEM>MAXNUMITEMS THEN
WRITE("+")
WRITELN
END
WRITELN(I:3)
END
READLN
END
END
END
```

# APPENDIX H

## CODE FOR PROGRAM ACCUSE



```
000004  0
000004  0     (*$R-*)
000004  0     PROGRAM  ACCUSE (INPUT,OUTPUT);
000074  0
000074  0     (*  SAM GRIER
000074  0         THESIS
000074  0
000074  0         EENIE, MEENIE, MINIE, MO, CATCH A CHEATER BY THE TOE...   *)
000074  0
000074  0
000074  0     LABEL     1;    (* TERMINATES THE PROGRAM AT EOF *)
000074  0
000074  0     CONST
000074  0             NUMKEYWORDS  =  156;  (* NUMBER OF KEYWORDS IN PASCAL *)
000074  0             MAXCHAR      =   63;  (* MAXIMUM CHARACTER ORDINAL *)
000074  0             QUOTE        = '''';  (* STRING QUOTE *)
000074  0             MAXSTRING    =  120;  (* LENGTH OF ANY LINE OF INPUT *)
000074  0             LNLIM        =  121;  (* MAXSTRING + 1 *)
000074  0             MAXNESTLEVEL =   50;  (* MAXIMUM NEST OF PROCEDURES *)
000074  0             MAXMODULE    =  140;  (* MAXIMUM NMBR OF PROGRAMS THAT CAN BE PROCESSED PLUS ONE *)
000074  0             CCIDLNGTH    =    4;  (* THE LENGTH OF THE COMPUTING CENTER I.D. *)
000074  0             PARACOUNT    =   20;  (* NMBR OF PARAMETERS BEING COUNTED *)
000074  0             SIGINDENT    =    2;  (* SIGNIFICANT INDENTATION *)
000074  0             CCIDPERPAGE  =   50;  (* NUMBER OF COUNTS PRINTED PER PAGE *)
000074  0
000074  0     TYPE
000074  0             COUNTCLASS   = (
000074  0                            EPLIN,   (* LINES IN PROGRAM *)
000074  0                            EPOPS,   (* OPERATORS IN ENTIRE PROGRAM *)
000074  0                            EPCT,    (* TYPES AND CONSTANTS IN ENTIRE PROGRAM *)
000074  0                            UNOPN,   (* UNIQUE OPERANDS IN PROGRAM *)
000074  0                            EPVAR,   (* VARIABLES IN ENTIRE PROGRAM *)
000074  0                            VARPA,   (* NUMBER OF VAR PARAMETERS *)
000074  0                            PRVAR,   (* NUMBER OF VARIABLES IN PROCEDURES *)
000074  0                            EPPRO,   (* NUMBER OF PROCEDURES AND FUNCTIONS *)
000074  0                            EPOPN,   (* OPERANDS IN ENTIRE PROGRAM *)
000074  0                            FORCT,   (* NUMBER OF FOR STMTS *)
000074  0                            GOCNT,   (* NUMBER OF GOTO STMTS *)
000074  0                            REPCT,   (* NUMBER OF REPEAT STMTS *)
000074  0                            WHICT,   (* NUMBER OF WHILE STMTS *)
000074  0                            EPCOM,   (* LINES OF COMMENTS IN THE PROGRAM *)
000074  0                            EPMOL,   (* LINES WITH MORE THAN ONE STMT *)
000074  0                            VALPA,   (* NUMBER OF VAL PARAMETERS *)
000074  0                            EPCL,    (* CODE LINES IN PROGRAM *)
000074  0                            INDNF,   (* RELATIVE INDENTING FUNCTION *)
000074  0                            UNOPS,   (* UNIQUE OPERATORS IN PROGRAM *)
000074  0                            VARNU);  (* VARIABLES DECLARED BUT NOT USED *)
000074  0
000074  0             TOKTYP       = (
000074  0                            LBLDEC,  (* PROCESS A LABEL DECL *)
000074  0                            CONDEC,  (* PROCESS A CONSTANT DECL *)
000074  0                            TYPDEC,  (* PROCESS A TYPE DECL *)
000074  0                            VARDEC,  (* PROCESS A VAR DECL *)
000074  0                            VALDEC,  (* PROCESS A VALUE DECL *)
000074  0                            PROG,    (* PROCESS THE PROGRAM DECL *)
000074  0                            PROCS,   (* PROCESS A PROCEDURE OR FUNCTION DECL *)
000074  0                            FWD,     (* PROCESS A FORWARD DECL *)
```

```
000074  0                           CASEST,   (* PROCESS A CASE STMT *)
000074  0                           FINDOF,   (* PROCESS AN OF *)
000074  0                           FINDBEG,  (* PROCESS A BEGIN *)
000074  0                           FINDEND,  (* PROCESS AN END *)
000074  0                           LPAR,     (* PROCESS LEFT PAREN *)
000074  0                           RPAR,     (* TERMINATE LEFT PAREN *)
000074  0                           FORST,    (* PROCESS A FOR STMT *)
000074  0                           GOTOST,   (* PROCESS A GOTO STMT *)
000074  0                           INST,     (* PROCESS AN IN STMT *)
000074  0                           REPST,    (* PROCESS A REPEAT STMT *)
000074  0                           WHIST,    (* PROCESS A WHILE STMT *)
000074  0                           COMNT,    (* PROCESS A COMMENT *)
000074  0                           LBRAC,    (* PROCESS AN ARRAY SUBSCRIPT *)
000074  0                           SEMI,     (* PROCESS A SEMI-COLON *)
000074  0                           OPND,     (* PROCESS AN OPERAND *)
000074  0                           NMBR,     (* PROCESS A NUMBER *)
000074  0                           BLNKS,    (* PROCESS ONE OR MORE BLANKS *)
000074  0                           DOLLAR,   (* INCORRECT $ POSITION *)
000074  0                           EOLNM,    (* PROCESS AN EOLN *)
000074  0                           EOLNSM,   (* PROCESS AN EOLNS *)
000074  0                           EOFM,     (* PROCESS AN EOF *)
000074  0                           EOFSM,    (* PROCESS AN EOFS *)
000074  0                           TERM,     (* TERMINATE CURRENT PROGRAM *)
000074  0                           ENDLIST,  (* TERMINATE ACCUSE *)
000074  0                           ABT,      (* ABORT CURRENT PROGRAM *)
000074  0                           FUNCS,    (* PROCESS A FUNCTION *)
000074  0                           RECDEC,   (* PROCESS A RECORD DECL *)
000074  0                           TOST,     (* PROCESS A TO STMT *)
000074  0                           WITHST,   (* PROCESS A WITH STMT *)
000074  0                           RBRAC,    (* TERMINATE ARRAY SUBSCRIPT *)
000074  0                           EQUAL,    (* PROCESS = *)
000074  0                           COLON,    (* PROCESS A COLON *)
000074  0                           ASSIGN,   (* PROCESS ASSIGNMT STMT *)
000074  0                           COMMA,    (* PROCESS A COMMA *)
000074  0                           DUMMY);   (* DO NOTHING *)
000074  0
000074  0          SYMLINK      = ^RESSYMBOL;(* POINTER TO RESERVED WORDS *)
000074  0
000074  0          RESSYMBOL    = RECORD
000074  1                           STRING:   ALFA;       (* KEYWORD *)
000074  1                           LNGTH:    INTEGER;    (* LENGTH OF WORD *)
000074  1                           OP:       BOOLEAN;    (* TRUE IF OPERATOR *)
000074  1                           USED:     BOOLEAN;    (* TRUE IF OPERATOR USED *)
000074  1                           NEXTSYM:  SYMLINK;    (* NEXT SYMBOL IN TABLE *)
000074  1                           LASTSYM:  SYMLINK;    (* PREVIOUS SYMBOL IN TABLE *)
000074  1                           NEXTIMP:  SYMLINK;    (* USER FUNCTION LINKAGE PTR *)
000074  1                           TOK:      TOKTYP;     (* TOKEN TYPE *)
000074  1       (* RESSYMBOL *)     END;
000074  0
000074  0          CHBUFF       = ARRAY [1..10] OF CHAR:  (* UTILITY BUFFER DECL *)
000074  0          TEXTNDX      = 1..MAXSTRING;           (* ALLOWABLE IDENBUFF INDICES *)
000074  0          MAXTEXT      = 1..LNLIM;               (* ALLOWABLE LINE INDICES *)
000074  0          MODULENUMBERS= 1..MAXMODULE;           (* POSSIBLE MODULE NUMBERS *)
000074  0
000074  0     VAR
000074  0         KEYWORDS:            (* KEYWORDS FOR SYMBOL TABLE INITIALIZATION *)
000074  0                     ARRAY [1..NUMKEYWORDS] OF RESSYMBOL;
```

```
002434  0
002434  0      SYMTBL:            (* SYMBOL TABLE THAT CONTAINS PASCAL KEYWORDS *)
002434  0                 ARRAY [CHAR] OF SYMLINK;
002534  0
002534  0      CHTBL:             (* SCANNER TABLE *)
002534  0                 ARRAY [CHAR] OF 0..12;
002634  0
002634  0      ACTION:    TOKTYP: (* ACTION TO BE PERFORMED BY THE DRIVER *)
002635  0      COUNTER:           (* PARAMETER TO BE COUNTED BY COUNT *)
002635  0                 COUNTCLASS;
002636  0
002636  0      IDENBUFF:          (* IDENTIFIER BUFFER *)
002636  0                 ARRAY [TEXTNDX] OF CHAR;
003026  0      ENDIDEN:   TEXTNDX: (* INDEX TO IDENBUFF *)
003027  0
003027  0      LINE:              (* INPUT LINE *)
003027  0                 ARRAY [MAXTEXT] OF CHAR;
003220  0      INDEX,             (* INDEX TO LINE *)
003220  0      ENDLINE:   MAXTEXT: (* LAST CHARACTER + 1 OF ANY LINE *)
003222  0
003222  0      LINESOFCOMMENT,    (* NUMBER OF LINES IN COMMENT BEING PROCESSED *)
003222  0
003222  0      NUMBLNKS:          (* NUMBER OF BLANKS BEING PROCESSED *)
003222  0                 INTEGER;
003224  0
003224  0      NEWLINE,           (* TRUE IF A LINE IS YET UNPROCESSED *)
003224  0      ISADELIMITER,      (* TRUE IF ITEM IS A DELIMITER *)
003224  0      KEYWORD:   BOOLEAN; (* TRUE IF AN IDENTIFIER IS A KEYWORD *)
003227  0
003227  0      PTRSYMTBL,         (* UTILITY POINTER FOR THE SYMBOL TABLE *)
003227  0      PTRNEWREC:         (* POINTER USED TO CREATE NEW RECORDS *)
003227  0                 SYMLINK;
003231  0
003231  0      AMTFWDDEC,         (* NUMBER OF FORWARD DECL *)
003231  0      STARTPOS:  INTEGER; (* INDEX FOR COMPUTING INDENFUNC *)
003233  0
003233  0      YES,               (* IF TRUE THEN DRIVER CALLS SCANNER *)
003233  0      SAMELINE,          (* SET TRUE AFTER FIRST STMT PER LINE *)
003233  0      INIT,              (* TRUE IF IN AN IN STMT *)
003233  0      FWDDEC,            (* TRUE IF PROCESSING A FORWARD DECL *)
003233  0      PRODECL,           (* TRUE IF PROCESSING A PROCEDURE DECL *)
003233  0      PARADEC:           (* TRUE IF PROCESSING FORMAL PARAMETERS *)
003233  0                 BOOLEAN;
003241  0      PRODECLNOTUSED:    (* TRUE IF ANY PROCEDURE IS DECLARED BUT NOT USED *)
003241  0                 ARRAY [MODULENUMBERS] OF BOOLEAN;
003455  0
003455  0      HEADNMBR:          (* HEAD OF THE LIST OF UNIQUE NUMBERS *)
003455  0                 SYMLINK;
003456  0
003456  0      NESTLEVEL:         (* POSSIBLE NESTLEVELS ARE 1..MAXNESTLEVEL *)
003456  0                 0..MAXNESTLEVEL;
003457  0      HEADREDEF,         (* LIST OF REDEFINED KEYWORDS AT EACH NESTLEVEL *)
003457  0      DECLIST,           (* LIST OF VARS AT EACH NESTLEVEL *)
003457  0      HEADTEMP:          (* LIST OF USER DEFINED FUNCTIONS AT EACH NESTLEVEL *)
003457  0                 ARRAY [1..MAXNESTLEVEL] OF SYMLINK;
003705  0      STARTPROC:         (* ENABLES PROGRAM TO TERMINATE PROCEDURES *)
003705  0                 ARRAY [1..MAXNESTLEVEL] OF INTEGER;
```

```
003767  0
003767  0        MODULE:              (* INDEX FOR PROGNOSIS AND CCID *)
003767  0                  0..MAXMODULE;
003770  0        PROGNOSIS:           (* ARRAY OF PARAMETERS OF INDIVIDUAL PROGRAMS *)
003770  0                  ARRAY [MODULENUMBERS,1..PARACOUNT] OF INTEGER;
011350  0        ORDER:               (* ARRAY OF SORTED MODULES *)
011350  0                  ARRAY [MODULENUMBERS] OF MODULENUMBERS;
011564  0        CCID:                (* CCID ARRAY *)
011564  0                  PACKED ARRAY [MODULENUMBERS,1..CCIDLNGTH] OF CHAR;
011672  0
011672  0        KEY,                 (* FINGERPRINT ELEMENT USED TO SORT PROGNOSIS *)
011672  0        NUMBEG,              (* CURRENT NUMBER OF BEGINS *)
011672  0        NUMCASE,             (* CURRENT NUMBER OF CASE STMTS *)
011672  0        TOTALLINES,          (* NUMBER OF TOTAL LINES IN PROGRAM *)
011672  0        OPERATORS,           (* NUMBER OF OPERATORS IN PROGRAM *)
011672  0        CONSANDTYPES,        (* NUMBER OF DIFFERENT CONSTANTS AND TYPES *)
011672  0        UNIQUEOPN,           (* UNIQUE OPERANDS IN PROGRAM *)
011672  0        VARIABLES,           (* VARIABLES IN THE PROGRAM *)
011672  0        VARPARA,             (* VAR PARAMETERS IN PROGRAM *)
011672  0        PROVAR,              (* PROCEDURE VARIABLES IN PROGRAM *)
011672  0        PROCANDFUNC,         (* PROCEDURES AND FUNCTIONS IN PROGRAM *)
011672  0        OPERANDS,            (* OPERANDS IN ENTIRE PROGRAM *)
011672  0        FORSTMT,             (* NUMBER OF FOR STMT IN PROGRAM *)
011672  0        GO,                  (* NUMBER OF GOTO STMT IN PROGRAM *)
011672  0        REPSTMT,             (* REPEAT STMTS IN PROGRAM *)
011672  0        WHISTMT,             (* WHILE STMTS IN PROGRAM *)
011672  0        TOTALCOMMENTS,       (* LINES OF COMMENTS IN A PROGRAM *)
011672  0        MORETHANONE,         (* LINES CONTAINING MORE THAT ONE STMT *)
011672  0        VALPARA,             (* VALUE PARAMETERS IN PROGRAM *)
011672  0        CODELINES,           (* CODE LINES IN PROGRAM *)
011672  0        VARNOTUSED,          (* VARIABLES DECL NOT USED *)
011672  0        UNIQUEOPS,           (* UNIQUE OPERATORS IN PROGRAM *)
011672  0        NONBLANKLINES:       (* NON BLANK LINES IN PROGRAM *)
011672  0                  INTEGER;
011721  0
011721  0        OUTLINE:  INTEGER; (* LINE NUMBER INDEX OF HEADING OUTPUT *)
011722  0
011722  0        LEFTINDENT,          (* NUMBER LEFT INDENTATIONS MADE FROM REFERENCE *)
011722  0        ZEROINDENT,          (* NUMBER ZERO INDENTATIONS MADE FROM REFERENCE *)
011722  0        RIGHTINDENT:         (* NUMBER RIGHT INDENTATIONS MADE FROM REFERENCE *)
011722  0                  INTEGER;
011725  0
011725  0        STAT:                (* ARRAY THAT IS USED TO DETERMINE PLAGIARISM *)
011725  0                  ARRAY [MODULENUMBERS,MODULENUMBERS] OF INTEGER;
060145  0
060145  0        ADDWASDONE,          (* TRUE IF ADD DONE FOR MORESORTS *)
060145  0        ENDFLAG:  BOOLEAN; (* NECESSARY FOR PROGRAM TERMINATION *)
060147  0
060147  0     VALUE
060147  0          (* INITIALIZE KEYWORDS FOR SYMBOL TABLE INITIALIZATION *)
060147  0
060147  0          KEYWORDS = ( ('ABS      ',3,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0                       ('ALFA     ',4,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                       ('AND      ',3,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                       ('ARCCOS   ',6,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0                       ('ARCSIN   ',6,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0                       ('ARCTAN   ',6,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
```

```
060147  0   229          ('ARCTAN2   ',7,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   230          ('ARRAY     ',5,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0   231          ('BEGIN     ',5,FALSE,FALSE,NIL,NIL,NIL,FINDBEG),
060147  0   232          ('BOOLEAN   ',7,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0   233          ('CARD      ',4,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   234          ('CASE      ',4,TRUE,FALSE,NIL,NIL,NIL,CASEST),
060147  0   235          ('CHAR      ',4,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0   236          ('CHR       ',3,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   237          ('CLOCK     ',5,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0   238          ('CONST     ',5,FALSE,FALSE,NIL,NIL,NIL,CONDEC),
060147  0   239          ('COS       ',3,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   240          ('COSH      ',4,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   241          ('DATE      ',4,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   242          ('DISPOSE   ',7,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   243          ('DIV       ',3,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0   244          ('DO        ',2,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0   245          ('DOUBLE    ',6,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0   246          ('DOWNTO    ',6,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0   247          ('DYNAMIC   ',7,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0   248          ('ELSE      ',4,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0   249          ('END       ',3,FALSE,FALSE,NIL,NIL,NIL,FINDEND),
060147  0   250          ('EOF       ',3,TRUE,FALSE,NIL,NIL,NIL,EOFM),
060147  0   251          ('EOFS      ',4,TRUE,FALSE,NIL,NIL,NIL,EOFSM),
060147  0   252          ('EOLN      ',4,TRUE,FALSE,NIL,NIL,NIL,EOLNM),
060147  0   253          ('EOLNS     ',5,TRUE,FALSE,NIL,NIL,NIL,EOLNSM),
060147  0   254          ('EOS       ',3,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   255          ('EOSS      ',4,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   256          ('EXP       ',3,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   257          ('EXPO      ',4,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   258          ('EXTERN    ',6,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0   259          ('FALSE     ',5,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0   260          ('FILE      ',4,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0   261          ('FOR       ',3,TRUE,FALSE,NIL,NIL,NIL,FORST),
060147  0   262          ('FORTRAN   ',7,FALSE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   263          ('FORWARD   ',7,FALSE,FALSE,NIL,NIL,NIL,FWD),
060147  0   264          ('FUNCTION  ',8,FALSE,FALSE,NIL,NIL,NIL,PROCS),
060147  0   265          ('GENSORT   ',7,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   266          ('GET       ',3,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   267          ('GETSEG    ',6,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   268          ('GETRAN    ',6,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   269          ('GETRANDOM ',9,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   270          ('GOTO      ',4,TRUE,FALSE,NIL,NIL,NIL,GOTOST),
060147  0   271          ('HALT      ',4,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0   272          ('HIGH      ',4,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0   273          ('IF        ',2,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0   274          ('IN        ',2,TRUE,FALSE,NIL,NIL,NIL,INST),
060147  0   275          ('INPUT     ',5,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0   276          ('INTEGER   ',7,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0   277          ('LABEL     ',5,FALSE,FALSE,NIL,NIL,NIL,LBLDEC),
060147  0   278          ('LINELIMIT ',9,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   279          ('LOG       ',3,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   280          ('LOGIO     ',5,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   281          ('LOW       ',3,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   282          ('LN        ',2,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   283          ('MAXINT    ',6,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0   284          ('MESSAGE   ',7,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147  0   285          ('MOD       ',3,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
```

```
060147 0   ('NEW       ',3,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('NIL       ',3,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147 0   ('NOT       ',3,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147 0   ('OCT       ',3,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147 0   ('ODD       ',3,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('OF        ',2,FALSE,FALSE,NIL,NIL,NIL,FINDOF),
060147 0   ('OR        ',2,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147 0   ('ORD       ',3,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('OTHERWISE ',9,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147 0   ('OUTPUT    ',6,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147 0   ('PACK      ',4,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('PACKED    ',6,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147 0   ('PAGE      ',4,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('POWER     ',5,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('POWERI    ',6,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('PRED      ',4,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('PROCEDURE ',9,FALSE,FALSE,NIL,NIL,NIL,PROCS),
060147 0   ('PROGRAM   ',7,FALSE,FALSE,NIL,NIL,NIL,PROG),
060147 0   ('PUT       ',3,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('PUTSEG    ',6,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('RAN       ',3,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('RANDOM    ',6,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('READ      ',4,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147 0   ('READLN    ',6,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147 0   ('READSTRING',10,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('REAL      ',4,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147 0   ('RELEASE   ',7,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('RECORD    ',6,FALSE,FALSE,NIL,NIL,NIL,RECDEC),
060147 0   ('REPEAT    ',6,TRUE,FALSE,NIL,NIL,NIL,REPST),
060147 0   ('RESET     ',5,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('REWRITE   ',7,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('ROUND     ',5,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('SEGMENTED ',9,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147 0   ('SET       ',3,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147 0   ('SETRAN    ',6,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('SETRANDOM ',9,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('SKIPBLANKS',10,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('SIN       ',3,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('SINH      ',4,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('SQR       ',3,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('SQRT      ',4,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('SUCC      ',4,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('TAN       ',3,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147 0   ('TANH      ',4,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147 0   ('TEXT      ',4,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147 0   ('THEN      ',4,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147 0   ('TIME      ',4,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('TO        ',2,FALSE,FALSE,NIL,NIL,NIL,TOST),
060147 0   ('TRUE      ',4,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147 0   ('TRUNC     ',5,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('TYPE      ',4,FALSE,FALSE,NIL,NIL,NIL,TYPDEC),
060147 0   ('UNPACK    ',6,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('UNTIL     ',5,FALSE,FALSE,NIL,NIL,NIL,DUMMY),
060147 0   ('UNDEFINED ',9,TRUE,FALSE,NIL,NIL,NIL,FUNCS),
060147 0   ('VALUE     ',5,FALSE,FALSE,NIL,NIL,NIL,VALDEC),
060147 0   ('VAR       ',3,FALSE,FALSE,NIL,NIL,NIL,VARDEC),
060147 0   ('WHILE     ',5,TRUE,FALSE,NIL,NIL,NIL,WHIST),
```

PASCAL COMPILER - E.T.H. ZUERICH / UNIVERSITY OF MINNESOTA.          PASCAL-6000 V3.2.0.  80/12/01. 19.35 08.
UNIVERSITY OF COLORADO COMPUTING CENTER                              KRONOS 2.1          (80/06/23)  PAGE 7

```
060147  0                    (*WITH      *,4,TRUE,FALSE,NIL,NIL,NIL,WITHST),
060147  0                    (*WRITE     *,5,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                    (*WRITELN   *,7,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                    (*+         *,1,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                    (*-         *,1,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                    (**         *,1,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                    (**)        *,2,FALSE,FALSE,NIL,NIL,NIL,ART),
060147  0                    (*/         *,1,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                    (*(         *,1,FALSE,FALSE,NIL,NIL,NIL,LPAR),
060147  0                    (*)         *,1,FALSE,FALSE,NIL,NIL,NIL,RPAR),
060147  0                    (*$         *,1,FALSE,FALSE,NIL,NIL,NIL,ART),
060147  0                    (*=         *,1,TRUE,FALSE,NIL,NIL,NIL,EQUAL),
060147  0                    (*,         *,1,TRUE,FALSE,NIL,NIL,NIL,COMMA),
060147  0                    (*..        *,2,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                    (*.         *,1,TRUE,FALSE,NIL,NIL,NIL,ISEMI),
060147  0                    (*[         *,1,TRUE,FALSE,NIL,NIL,NIL,LBRAC),
060147  0                    (*]         *,1,FALSE,FALSE,NIL,NIL,NIL,RBRAC),
060147  0                    (*:         *,1,FALSE,FALSE,NIL,NIL,NIL,COLON),
060147  0                    (*:=        *,2,TRUE,FALSE,NIL,NIL,NIL,ASSIGN),
060147  0                    (*<         *,1,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                    (*>         *,1,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                    (*<>        *,2,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                    (*;         *,1,FALSE,FALSE,NIL,NIL,NIL,SEMI),
060147  0                    (*<=        *,2,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                    (*>=        *,2,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                    (*         *,1,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                    (*#         *,1,FALSE,FALSE,NIL,NIL,NIL,ART),
060147  0                    (*&         *,1,FALSE,FALSE,NIL,NIL,NIL,ART),
060147  0                    (*?         *,1,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                    (*         *,1,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                    (*'         *,1,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                    (*".        *,2,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                    (*"         *,1,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                    (* .        *,2,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                    (*          *,1,TRUE,FALSE,NIL,NIL,NIL,DUMMY),
060147  0                    (*          *,1,TRUE,FALSE,NIL,NIL,NIL,DUMMY) );
060147  0          (* END KEYWORDS *)
060147  0
060147  0
060147  0
060147  0     PROCEDURE SCANNER;
000002  0     FORWARD;
000002  0
000002  0
000002  0
000002  0     PROCEDURE GETLINE;
000002  0     FORWARD;
000002  0
000002  0
000002  0
000002  0     PROCEDURE DRIVER;
000002  0     FORWARD;
000002  0
000002  0
000002  0
000002  0     PROCEDURE COUNT (COUNTER:COUNTCLASS);
000003  0     FORWARD;
```

```
000003  0
000003  0
000003  0
000003  0   PROCEDURE MODINIT;
000002  0   FORWARD;
000002  0
000002  0
000002  0
000002  0   PROCEDURE EXITNMBR;
000002  0   FORWARD;
000002  0
000002  0
000002  0
000002  0   PROCEDURE EXITPROCEDURE;
000002  0   FORWARD;
000002  0
000002  0
000002  0
000002  0   PROCEDURE INSERT;
000002  0     (* INSERT KEYWORDS INTO THE SYMBOL TABLE *)
000002  0
000002  0   VAR
000002  0        BUFF:       CHBUFF;
000014  0
000014  0   BEGIN (* INSERT *)
000014  1
000014  1   UNPACK (PTRNEWREC^.STRING,BUFF,1);
000014  1   IF  SYMTBL[BUFF[1]] = NIL
000018  1   THEN  SYMTBL[BUFF[1]] := PTRNEWREC
000022  1   ELSE  BEGIN
000024  2         PTRSYMTBL := SYMTBL[BUFF[1]];
000027  2         WHILE  PTRSYMTBL^.NEXTSYM <> NIL  DO
000035  2                PTRSYMTBL := PTRSYMTBL^.NEXTSYM;
000043  2         PTRSYMTBL^.NEXTSYM := PTRNEWREC;
000050  2         PTRNEWREC^.LASTSYM := PTRSYMTBL;
000055  2         END;
000055  1
000055  1   END;  (* INSERT *)
000057  0
000057  0
000057  0
000057  0   PROCEDURE SYMINIT;
000002  0     (* INITIALIZE THE SYMBOL TABLE *)
000002  0
000002  0   VAR
000002  0       C:           CHAR;
000003  0       I:           INTEGER;
000004  0
000004  0   BEGIN (* SYMINIT *)
000004  1
000004  1   FOR C := CHR(0) TO CHR(MAXCHAR) DO
000007  1       SYMTBL[C] := NIL;
000020  1
000020  1   FOR I := 1 TO NUMKEYWORDS DO
000021  1       BEGIN
000023  2       NEW (PTRNEWREC);
000025  2       PTRNEWREC^ := KEYWORDS[I];
```

```
000040  2       INSERT;
000041  2       END;
000046  1
000046  1   END;  (* SYMINIT *)
000054  0
000054  0
000054  0
000054  0   PROCEDURE SCANINIT;
000002  0     (* INITIALIZE THE SCANNER *)
000002  0
000002  0   VAR
000002  0       C:       CHAR;
000003  0
000003  0   BEGIN (* SCANINIT *)
000003  1
000003  1   FOR C := CHR(0) TO CHR(MAXCHAR) DO
000007  1       CHTBL[C] := 12;
000020  1
000020  1   CHTBL["0"] :=  0;
000021  1   CHTBL["1"] :=  1;
000022  1   CHTBL["2"] :=  2;
000023  1   CHTBL["3"] :=  3;
000025  1   CHTBL["4"] :=  4;
000026  1   CHTBL["5"] :=  5;
000030  1   CHTBL["6"] :=  6;
000031  1   CHTBL["7"] :=  7;
000033  1   CHTBL["8"] :=  8;
000034  1   CHTBL["9"] :=  9;
000036  1   CHTBL["A"] := 10;
000037  1   CHTBL["B"] := 10;
000041  1   CHTBL["C"] := 10;
000042  1   CHTBL["D"] := 10;
000044  1   CHTBL["E"] := 10;
000045  1   CHTBL["F"] := 10;
000047  1   CHTBL["G"] := 10;
[illegible]  1   CHTBL["H"] := 10;
[illegible]  1   CHTBL["I"] := 10;
[illegible]  1   CHTBL["J"] := 10;
[illegible]  1   CHTBL["K"] := 10;
[illegible]  1   CHTBL["L"] := 10;
[illegible]  1   CHTBL["M"] := 10;
[illegible]  1   CHTBL["N"] := 10;
000063  1   CHTBL["O"] := 10;
000064  1   CHTBL["P"] := 10;
[illegible]  1   CHTBL["Q"] := 10;
[illegible]  1   CHTBL["R"] := 10;
[illegible]  1   CHTBL["S"] := 10;
[illegible]  1   CHTBL["T"] := 10;
[illegible]  1   CHTBL["U"] := 10;
000075  1   CHTBL["V"] := 10;
[illegible]  1   CHTBL["W"] := 10;
[illegible]  1   CHTBL["X"] := 10;
[illegible]  1   CHTBL["Y"] := 10;
[illegible]  1   CHTBL["Z"] := 10;
[illegible]  1   CHTBL[QUOTE] := 11;
[illegible]  1
[illegible]  1   END; (* SCANINIT *)
```

```
000112  0
000112  0
000112  0
000112  0   PROCEDURE DUMPSYMTBL;
000002  0      (* DUMP THE CURRENT CONTENTS OF THE SYMBOL TABLE.
000002  0         THE FOLLOWING VALUES ARE DUMPED:
000002  0             STRING
000002  0             OP
000002  0             USED *)
000002  0
000002  0    VAR  C: CHAR;
000003  0
000003  0    BEGIN (* DUMPSYMTBL *)
000003  1
000003  1    WRITELN ('1');
000012  1    FOR  C := CHR(0) TO CHR(MAXCHAR)  DO
000015  1         IF  SYMTBL[C] <> NIL
000022  1         THEN  BEGIN
000024  2               WRITELN('0');
000032  2               PTRSYMTBL := SYMTBL[C];
000036  2               WHILE  PTRSYMTBL <> NIL  DO
000041  2                      WITH  PTRSYMTBL^  DO
000045  2                            BEGIN
000045  3                            WRITE (' KEYWORD = ',STRING:10);
000057  3                            WRITE ('     OP = ', OP:5);
000070  3                            WRITELN ('     USED = ', USED:5);
000102  3                            PTRSYMTBL := NEXTSYM;
000105  3                            END;
000106  2               END;
000113  1
000113  1    END;  (* DUMPSYMTBL *)
000124  0
000124  0
000124  0
000124  0    PROCEDURE ABORT;
000002  0       (* ABORT THE PRESENT PROGRAM-
000002  0          IT IS UNCOMPILEABLE *)
000002  0
000002  0    VAR   I:   INTEGER;
000003  0
000003  0    BEGIN (* ABORT *)
000003  1
000003  1    EXITNMBR;
000005  1    WHILE  NESTLEVEL > 0  DO
000007  1           EXITPROCEDURE;
000011  1    COUNT (UNOPS);
000015  1    WHILE  (ACTION <> TERM) AND (ENDFLAG <> TRUE)  DO
000022  1           BEGIN
000022  2           NEWLINE := FALSE;
000023  2           SCANNER;
000024  2           END;
000025  1    (* INSERT THE LINENUMBER WHERE THE ABORT OCCURRED *)
000025  1    PROGNOSIS[MODULE,1] := TOTALLINES;
000033  1    FOR I := 2 TO PARACOUNT DO
000034  1        PROGNOSIS[MODULE,I] := 0;
000050  1    MODINIT;
000051  1
```

```
000051  1   END;  (* ABORT *)
000055  0
000055  0
000055  0
000055  0   FUNCTION TEST: BOOLEAN;
000003  0      (* SEE IF THE INPUT STRING IS A KEYWORD
000003  0         ON ENTRY-
000003  0            IDENBUFF[1..ENDIDEN] = STRING TO BE CHECKED
000003  0         ON EXIT-
000003  0            IF STRING IS NOT A KEYWORD THEN TEST = FALSE
000003  0            IF STRING IS A KEYWORD THEN TEST = TRUE *)
000003  0
000003  0   LABEL 1,2;
000003  0
000003  0   VAR   BUFF: CHBUFF;
000015  0         I:    TEXTNDX;
000016  0
000016  0   BEGIN (* TEST *)
000016  1
000016  1   TEST := FALSE;
000010  1   PTRSYMTBL := SYMTBL[IDENBUFF[1]];
000014  1   WHILE  PTRSYMTBL <> NIL  DO
000017  1          BEGIN
000017  2          IF  ENDIDEN = PTRSYMTBL^.LNGTH
000023  2          THEN  WITH PTRSYMTBL^ DO
000031  2                     BEGIN
000031  3                     UNPACK (STRING,BUFF,1);
000035  3                     FOR I := 1 TO ENDIDEN DO
000037  3                           IF  IDENBUFF[I] <> BUFF[I]
000047  3                           THEN  GOTO 1;
000057  3                     TEST := TRUE;
000060  3                     GOTO 2;
000061  3                     END;
000061  2          1:  PTRSYMTBL := PTRSYMTBL^.NEXTSYM;
000067  2          END;
000070  1
000070  1   2: END; (* TEST *)
000075  0
000075  0
000075  0
000075  0   PROCEDURE GETLINE;
000002  0      (* READ A LINE OF TEXT INTO A BUFFER
000002  0         ON EXIT-
000002  0            LINE[1] = FIRST CHARACTER OF THE LINE
000002  0            INDEX = 1
000002  0            LINE[ENDLINE] = QUOTE TO TERMINATE LINE
000002  0            ALL BLANKS REMAIN IN THE LINE *)
000002  0
000002  0   VAR  I:  INTEGER;
000003  0
000003  0   BEGIN (* GETLINE *)
000003  1
000003  1   IF  EOP
000004  1   THEN  BEGIN
000006  2         ACTION := ENDLIST;
000007  2         KEYWORD := FALSE;
000011  2         YES := FALSE;
```

```
000012  2                DRIVER;
000013  2                END
000013  1         ELSE   BEGIN
000014  2                ENDLINE := 1;
000016  2                WHILE  (NOT EOLN) AND (ENDLINE < LNLIM)  DO
000021  2                      BEGIN
000021  3                      READ (LINE[ENDLINE]);
000030  3                      ENDLINE := ENDLINE + 1;
000034  3                      END;
000035  2                READLN;
000037  2                INDEX := 1;
000041  2                LINE[ENDLINE] := QUOTE;
000045  2                NEWLINE := TRUE;
000046  2                SAMELINE := FALSE;
000050  2                END;
000050  1
000050  1         COUNT (EPLIN);
000053  1
000053  1         END;  (* GETLINE *)
000057  0
000057  0
000057  0
000057  0         PROCEDURE SCANNER;
000002  0            (* OBTAIN THE NEXT TOKEN FROM THE INPUT LINE AND PASS IT TO THE DRIVER
000002  0                ON ENTRY-
000002  0                   LINE[INDEX] = FIRST UNEXAMINED CHAR
000002  0                   LINE[ENDLINE] = QUOTE ACTING AS LINE TERMINATOR
000002  0                ON EXIT-
000002  0                   TOKTYP DESCRIBES THE TOKEN OBTAINED
000002  0                   IF KEYWORD = FALSE THEN ACTION CONTAINS TOKTYP
000002  0                   IF KEYWORD = TRUE THEN PTRSYMTBL^.TOK CONTAINS TOKTYP.
000002  0                THIS SCANNER IS A MODIFICATION OF THE PASCALJ COMPILER SCANNER
000002  0                WRITTEN AT THE UNIVERSITY OF COLORADO *)
000002  0
000002  0
000002  0                    PROCEDURE IDENTIFIER;
000002  0                       (* EXTRACT AN IDENTIFIER OR KEYWORD FROM THE INPUT LINE
000002  0                          ON ENTRY-
000002  0                             LINE[INDEX] CONTAINS A LETTER
000002  0                          ON EXIT-
000002  0                             LINE[INDEX] CONTAINS THE CHARACTER FOLLOWING THE IDENTIFIER
000002  0                             IF KEYWORD = FALSE THEN WE HAVE AN IDENTIFIER AND ACTION
000002  0                             CONTAINS TOKTYP
000002  0                             IF KEYWORD = TRUE THEN WE HAVE A KEYWORD AND PTRSYMTBL^.TOK
000002  0                             CONTAINS TOKTYP *)
000002  0
000002  0                    BEGIN (* IDENTIFIER *)
000002  1
000002  1                    IDENBUFF[1] := LINE[INDEX];
000013  1                    ENDIDEN := 1;
000015  1                    INDEX := INDEX + 1;
000017  1                    WHILE  CHTOCL[LINE[INDEX]] < 11  DO
000022  1                          BEGIN
0000..  2                          ENDIDEN := ENDIDEN + 1;
0000..  2                          IDENBUFF[ENDIDEN] := LINE[INDEX];
0000..  2                          INDEX := INDEX + 1;
0000..  2                          END;
```

```
000043  1   685            IF  ENDIDEN > 10
000043  1   686            THEN  ENDIDEN := 10;
000047  1   687            KEYWORD := TEST;
000053  1   688            IF  NOT KEYWORD
000053  1   689            THEN  ACTION := OPND
000054  1   690            ELSE  ACTION := PTRSYMTBL^.TOK;
000066  1   691            NEWLINE := FALSE;
000070  1   692
000070  1   693            END;  (* IDENTIFIER *)
000072  0   694
000072  0   695
000072  0   696            PROCEDURE NUMBER;
000002  0   697               (* EXTRACT A NUMBER FROM THE INPUT LINE
000002  0   698                  ON ENTRY-
000002  0   699                     LINE[INDEX] CONTAINS A DIGIT
000002  0   700                  ON EXIT-
000002  0   701                     LINE[INDEX] CONTAINS THE CHARACTER FOLLOWING THE NUMBER
000002  0   702                     ACTION CONTAINS TOKTYP *)
000002  0   703
000002  0   704               PROCEDURE SCANDIG;
000002  0   705                  (* SCANNER FOR A DIGIT STRING *)
000002  0   706
000002  0   707               BEGIN (* SCANDIG *)
000002  1   708
000002  1   709               WHILE  CHTBL[LINE[INDEX]] < 10  DO
000013  1   710                      BEGIN
000013  2   711                      ENDIDEN := ENDIDEN + 1;
000017  2   712                      IDENBUFF[ENDIDEN] := LINE[INDEX];
000025  2   713                      INDEX := INDEX + 1;
000030  2   714                      END;
000031  1   715
000031  1   716               END;  (* SCANDIG *)
000033  0   717
000033  0   718             BEGIN (* NUMBER *)
000033  1   719
000033  1   720             IDENBUFF[1] := LINE[INDEX];
000012  1   721             INDEX := INDEX + 1;
000015  1   722             ENDIDEN := 1;
000016  1   723             SCANDIG;
000020  1   724             IF  (LINE[INDEX] = ".") AND (CHTBL[LINE[INDEX + 1]] < 10)
000032  1   725             THEN  BEGIN
000033  2   726                   ENDIDEN := ENDIDEN + 1;
000037  2   727                   IDENBUFF[ENDIDEN] := LINE[INDEX];
000046  2   728                   INDEX := INDEX + 1;
000050  2   729                   SCANDIG;
000052  2   730                   END;
000052  1   731             IF  LINE[INDEX] = "E"
000055  1   732             THEN  BEGIN
000057  2   733                   ENDIDEN := ENDIDEN + 1;
000063  2   734                   IDENBUFF[ENDIDEN] := "E";
000066  2   735                   INDEX := INDEX + 1;
000071  2   736                   IF  (LINE[INDEX] = "+") OR (LINE[INDEX] = "-")
000077  2   737                   THEN  BEGIN
000101  3   738                         ENDIDEN := ENDIDEN + 1;
000104  3   739                         IDENBUFF[ENDIDEN] := LINE[INDEX];
000113  3   740                         INDEX := INDEX + 1;
000116  3   741                         END;
```

```
000118  2              SCANDIG;
000120  2              END;
000120  1          IF  ENDIDEN > 10
000120  1          THEN  ENDIDEN := 10;
000124  1          ACTION := NMBR;
000126  1          NEWLINE := FALSE;
000127  1
000127  1          END;  (* NUMBER *)
000131  0
000131  0          PROCEDURE STRING;
000002  0             (* EXTRACT A STRING FROM THE INPUT LINE
000002  0                ON ENTRY-
000002  0                   LINE[INDEX] = QUOTE
000002  0                   LINE[ENDLINE] = QUOTE ACTING AS LINE TERMINATOR
000002  0                   INDEX < ENDLINE
000002  0                ON EXIT-
000002  0                   LINE[INDEX] CONTAINS THE CHARACTER FOLLOWING THE STRING
000002  0                   ACTION CONTAINS TOKTYP *)
000002  0
000002  0                   PROCEDURE SCANSEG;
000002  0                      (* SCANNER FOR A SEGMENT OF A STRING *)
000002  0
000002  0                   BEGIN (* SCANSEG *)
000002  1
000002  1                   INDEX := INDEX + 1;
000010  1                   WHILE  LINE[INDEX] <> QUOTE  DO
000015  1                            INDEX := INDEX + 1;
000020  1                   IF  INDEX < ENDLINE
000020  1                   THEN  INDEX := INDEX + 1
000022  1                   ELSE  ABORT;
000026  1
000026  1                   END;  (* SCANSEG *)
000030  0
000030  0          BEGIN (* STRING *)
000030  1
000030  1          SCANSEG;
000006  1          WHILE  (INDEX < ENDLINE) AND (LINE[INDEX] = QUOTE)  DO
000015  1                 SCANSEG;
000017  1          ACTION := OPND;
000021  1          NEWLINE := FALSE;
000022  1
000C..  1          END;  (* STRING *)
0000..  0
000024  0          PROCEDURE DELIMITER;
000002  0             (* EXTRACT A DELIMITER FROM THE INPUT LINE
000002  0                ON ENTRY-
00000.  0                   LINE[INDEX] CONTAINS THE FIRST CHARACTER OF THE DELIMITER
0000C.  0                   INDEX < ENDLINE
0000C.  0                ON .(IT-
C000.   0                   LINE[INDEX] CONTAINS THE CHARACT.? FOLLOWING THE DEL.MITER
0000C.  0                   .RSYMIBL..TOR CONTAINS TOKTYP *)
00000.  0
0020C.  0                   PROCEDURE COMMENT;
00000.  0                      (* PROC..S A COMMENT
0000.   0                         ON E..RY-
0000..  0                            LI.E[INDEX..I.DEX+1] . . ..
0000..  0                         ON E...-
```

```
000002  0              LINE[INDEX] = CHARACTER FOLLOWING THE COMMENT *)
000002  0
000002  0         LABEL 1;
000002  0
000002  0         BEGIN (* COMMENT *)
000002  1
000002  1         LINESOFCOMMENT := 0;
000006  1         INDEX := INDEX + 2;
000012  1         LINE[ENDLINE] := "*";
000015  1         WHILE  TRUE  DO
000017  1              BEGIN
000017  2              WHILE  LINE[INDEX] <> "*"  DO
000024  2                     INDEX := INDEX + 1;
000027  2              IF  INDEX < ENDLINE
000027  2              THEN  BEGIN
000031  3                    INDEX := INDEX + 1;
000033  3                    IF  LINE[INDEX] = ")"
000035  3                    THEN  BEGIN
000037  4                          LINE[ENDLINE] := QUOTE;
000042  4                          LINESOFCOMMENT := LINESOFCOMMENT + 1;
000044  4                          INDEX := INDEX + 1;
000047  4                          GOTO 1;
000050  4                          END;
000050  3                    END
000050  2              ELSE  BEGIN
000051  3                    LINESOFCOMMENT := LINESOFCOMMENT + 1;
000053  3                    GETLINE;
000054  3                    LINE[ENDLINE] := "*";
000059  3                    END;
000060  2               END;
000061  1         1: ACTION := COMNT;
000063  1         NEWLINE := FALSE;
000064  1
000064  1         END;  (* COMMENT *)
000066  0
000066  0    BEGIN (* DELIMITER *)
000066  1
000066  1    IF  (LINE[INDEX] = "(") AND (LINE[INDEX + 1] = "*")
000014  1    THEN  COMMENT
000015  1    ELSE  BEGIN
000020  2          ISADELIMITER := TRUE;
000022  2          IDENBUFF[1] := LINE[INDEX];
000027  2          IDENBUFF[2] := LINE[INDEX + 1];
000033  2          ENDIDEN := 2;
000035  2          KEYWORD := TEST;
000041  2          IF  KEYWORD
000041  2          THEN  INDEX := INDEX + 2
000042  2          ELSE  BEGIN
000046  3                ENDIDEN := 1;
000050  3                KEYWORD := TEST;
000054  3                INDEX := INDEX + 1;
000057  3                END;
000057  2          ACTION := PIRSYMBL^.TOK;
000057  2          END;
000067  1    NEWLINE := FALSE;
000071  1
000071  1    END;  (* DELIMITER *)
```

```
000073 0
000073 0          PROCEDURE BLANKS;
000002 0             (* EXTRACT ONE OR MORE BLANKS FROM THE INPUT LINE
000002 0                ON EXIT-
000002 0                   LINE[INDEX] POINTS TO THE FIRST CHARACTER FOLLOWING
000002 0                   THE BLANK(S)
000002 0                   ACTION CONTAINS TOKTYP *)
000002 0
000002 0          BEGIN (* BLANKS *)
000002 1
000002 1          NUMBLNKS := 0;
000006 1          WHILE  LINE[INDEX] = ' '  DO
000013 1                 BEGIN
000013 2                 NUMBLNKS := NUMBLNKS + 1;
000015 2                 INDEX := INDEX + 1;
000017 2                 END;
000020 1          ACTION := BLNKS;
000022 1
000022 1          END;  (* BLANKS *)
000024 0
000024 0  BEGIN (* SCANNER *)
000024 1
000024 1  KEYWORD := FALSE;
000006 1  ISADELIMITER := FALSE;
000007 1  IF  (NOT NEWLINE) AND (INDEX = ENDLINE)
000011 1  THEN  GETLINE;
000014 1  IF NEWLINE
000014 1  THEN  BLANKS
000016 1  ELSE  IF  LINE[INDEX] = ' '
000023 1        THEN  BLANKS
000025 1        ELSE  CASE CHTBL[LINE[INDEX]] OF
000035 2                 0,1,2,3,4,5,6,7,8,9:  NUMBER;
000040 2                 10:  IDENTIFIER;
000043 2                 11:  STRING;
000046 2                 12:  DELIMITER;
000051 2              END; (* CASE *)
000053 1
000063 1  END;  (* SCANNER *)
000065 0
000065 0
000065 0
000065 0  PROCEDURE ACCUSEINIT;
000002 0     (* INITIALIZE VARIABLES FOR ACCUSE *)
000002 0
000002 0  VAR   I,J:   INTEGER;
000004 0
000004 0  BEGIN (* ACCUSEINIT *)
000004 1
000004 1  (* INITIALIZE STAT FOR ACCUSE STP *)
000004 1  FOR I := 1 TO MAXMODULE DO
000006 1      FOR J := 1 TO MAXMODULE DO
000012 1          STAT[I,J] := 0;
000031 1
000031 1  (* INITIALIZE ORDER FOR SORTPROGNOSIS *)
000031 1  FOR I := 1 TO MAXMODULE DO
000032 1      ORDER[I] := 1;
000044 1
```

```
000044  1   (* INITIALIZE MODULE *)
000044  1   MODULE := 0;
000045  1
000045  1   (* INITIALIZE HEAD OF NMBR LIST *)
000045  1   HEADNMBR := NIL;
000047  1
000047  1   ADDWASDONE := FALSE;
000050  1
000050  1   MODINIT;
000051  1
000051  1   END;  (* ACCUSEINIT *)
000057  0
000057  0
000057  0
000057  0   PROCEDURE MODINIT;
000002  0      (* INITIALIZE VARIABLES FOR EACH MODULE *)
000002  0
000002  0
000002  0   BEGIN (* MODINIT *)
000002  1
000002  1   MODULE := MODULE + 1;
000010  1   STARTPOS      := 0;
000011  1   LEFTINDENT    := 0;
000013  1   ZEROINDENT    := 0;
000014  1   RIGHTINDENT   := 0;
000015  1   NUMBLNKS      := 0;
000016  1   NUMBEG        := 0;
000020  1   NUMCASE       := 0;
000021  1   AMIF#DDEC     := 0;
000022  1
000022  1   TOTALLINES          := 0;
000023  1   OPERATORS           := 0;
000024  1   CONSANDTYPES        := 0;
000025  1   UNIQUEOPN           := 0;
000026  1   VARIABLES           := 0;
000027  1   VARPARA             := 0;
000030  1   PROVAR              := 0;
000031  1   PROCANDFUNC         := 0;
000032  1   OPERANDS            := 0;
000033  1   FORSTMT             := 0;
000034  1   GO                  := 0;
000035  1   REPSTMT             := 0;
000036  1   WHISTMT             := 0;
000037  1   TOTALCOMMENTS       := 0;
000040  1   MORETHANONE         := 0;
000041  1   VALPARA             := 0;
000042  1   CODELINES           := 0;
000043  1   VARNOTUSED          := 0;
000044  1   UNIQUEOPS           := 0;
000045  1
000045  1   YES                 := TRUE;
000047  1   ENDFLAG             := FALSE;
000050  1   PARADEC             := FALSE;
000052  1   PRODECL             := FALSE;
000053  1   FWDDEC              := FALSE;
000054  1   PRODECLNOTUSED[MODULE]    := FALSE;
000057  1
```

```
000057 1  IF  NOT EOF
000057 1  THEN  GETLINE;
000061 1
000061 1  END:  (* MODINIT *)
000063 0
000063 0
000063 0
000063 0  PROCEDURE COUNTUNOPS;
000002 0     (* TRAVERSE THE SYMBOL TABLE.  IF USED = TRUE THEN INCREMENT UNIQUEOPS *)
000002 0
000002 0  VAR  C:   CHAR;
000003 0
000003 0  BEGIN (* COUNTUNOPS *)
000003 1
000003 1  FOR C := CHR(0) TO CHR(MAXCHAR) DO
000007 1       BEGIN
000013 2       PTRSYMTBL := SYMTBL[C];
000016 2       WHILE  PTRSYMTBL <> NIL  DO
000021 2               WITH  PTRSYMTBL^  DO
000025 2                       BEGIN
000025 3                       IF  USED
000025 3                       THEN  BEGIN
000027 4                               UNIQUEOPS := UNIQUEOPS + 1;
000031 4                               USED := FALSE;
000032 4                               END;
000032 3                       PTRSYMTBL := NEXTSYM;
000035 3                       END;
000036 2       END;
000043 1
000043 1  END;  (* COUNTUNOPS *)
000047 0
000047 0
000047 0
000047 0  PROCEDURE DECLARE;
000002 0     (* PUT DECLARED VARIABLES INTO THE DECLIST AT THE APPROPRIATE NESTLEVEL *)
000002 0
000002 0  VAR      I:        INTEGER;
000003 0
000003 0  BEGIN (* DECLARE *)
000003 1
000003 1  IF  NESTLEVEL > 1
000004 1  THEN  COUNT (PRVAR);
000012 1  NEW (PTRNEWREC);
000014 1  WITH  PTRNEWREC^  DO
000021 1       BEGIN
000021 2       FOR I := 1 TO ENDIDEN DO
000024 2            STRING[I] := IDENBUFF[I];
000043 2       LNGTH := ENDIDEN;
000045 2       OP := FALSE;
000046 2       USED := FALSE;
000047 2       NEXTSYM := NIL;
000050 2       END;
000050 1  IF  DECLIST[NESTLEVEL] <> NIL
000053 1  THEN  BEGIN
000055 2       DECLIST[NESTLEVEL]^.LASTSYM := PTRNEWREC;
000065 2       PTRNEWREC^.NEXTSYM := DECLIST[NESTLEVEL];
000073 2       END;
```

```
000073  1   DECLIST[NESTLEVEL] := PTRNEWREC;
000100  1   PTRNEWREC^.LASTSYM := NIL;
000105  1
000105  1   END;  (* DECLARE *)
000111  0
000111  0
000111  0
000111  0   PROCEDURE OPNUSED;
000002  0      (* ON ENTRY-
000002  0            LIST OF VARIABLES DECLARED
000002  0         ON EXIT-
000002  0            LIST OF VARIABLES DECLARED MINUS CURRENT OPERAND *)
000002  0
000002  0   LABEL  1,2;
000002  0
000002  0   VAR     I,J:    INTEGER;
000004  0           PTR:    SYMLINK;
000005  0           BUFF:   CHBUFF;
000017  0
000017  0   BEGIN (* OPNUSED *)
000017  1
000017  1   FOR I := NESTLEVEL DOWNTO 1 DO
000006  1        BEGIN
000010  2        PTR := DECLIST[I];
000014  2        WHILE  PTR <> NIL  DO
000017  2               BEGIN
000017  3               IF  ENDIDEN = PTR^.LNGTH
000023  3               THEN  WITH  PTR^  DO
000031  3                     BEGIN
000031  4                     UNPACK (STRING,BUFF,1);
000035  4                     FOR J := 1 TO ENDIDEN DO
000037  4                         IF  IDENBUFF[J] <> BUFF[J]
000046  4                         THEN  GOTO 1;
000056  4                     IF  I = 1
000056  4                     THEN  BEGIN
00      5                           IF  LASTSYM <> NIL
00      5                           THEN  LASTSYM^.NEXTSYM := NEXTSYM
00      5                           ELSE  DECLIST[I] := NEXTSYM;
0000    5                           IF  NEXTSYM <> NIL
0000    5                           THEN  NEXTSYM^.LASTSYM := LASTSYM;
00      5                           DISPOSE (PTR);
00      5                           END
0001    4                     ELSE  USED := TRUE;
000111  4                     GOTO 2;
000112  4                     END;
0001    3               1: PTR := PTR^.NEXTSYM;
000125  3               END;
0001    2        END;
000125  1   2:
00012   1   END;  (* OPNUSED *)
0001    0
0001    0
0001    0
0001    0   PROCEDURE INSERTFN (NESTLEVEL:INTEGER);
000001  0      (* INSERTS USER DEFINED FUNCTIONS INTO SYMTBL;  ON EXIT FROM THE
0000    0         PARTICULAR NESTLEVEL, THESE FUNCTIONS ARE DESTROYED BY EXITPRO-
0000    0         CEDURE *)
```

```
BEGIN (* INSERTTEMP *)

IF  HEADTEMP[NESTLEVEL] <> NIL
THEN  PTRNEWREC^.NEXTTMP := HEADTEMP[NESTLEVEL];
HEADTEMP[NESTLEVEL] := PTRNEWREC;
INSERT;

END;  (* INSERTTEMP *)

        .

PROCEDURE REDEF;
   (*  INSERT THE KEYWORD ONTO A LIST OF REDEFINED KEYWORDS AND REMOVE THE
       KEYWORD FROM THE SYMBOL TABLE *)

BEGIN (* REDEF *)

PTRSYMTBL^.NEXTTMP := HEADREDEF[NESTLEVEL];
HEADREDEF[NESTLEVEL] := PTRSYMTBL;
WITH  PTRSYMTBL^  DO
      BEGIN
      IF  LASTSYM <> NIL
      THEN  LASTSYM^.NEXTSYM := NEXTSYM
      ELSE  SYMTBL[STRING[1]] := NEXTSYM;
      IF  NEXTSYM <> NIL
      THEN  NEXTSYM^.LASTSYM := LASTSYM;
      NEXTSYM := NIL;
      LASTSYM := NIL;
      END;

END;  (* REDEF *)



PROCEDURE EXITPROCEDURE;
   (* DECREMENT THE NESTLEVEL AND COUNT THE NUMBER OF VARIABLES AND FUNCTION
      PARAMETERS DECLARED AND NOT USED IN THE PROCEDURE *)

LABEL 1;
VAR   PTRVARS:   SYMLINK;

BEGIN (* EXITPROCEDURE *)

IF  NESTLEVEL <= 0
THEN  GOTO 1;

(* COUNT VARIABLES DECLARED AND NOT USED *)
PTRVARS := DECLIST[NESTLEVEL];
WHILE  PTRVARS <> NIL  DO
        WITH  PTRVARS^  DO
              BEGIN
              IF  NOT USED
              THEN  BEGIN
                    IF  NOT FWDDEC
                    THEN  COUNT (VARNU);
                    IF  NESTLEVEL > 1
```

```
000032  3  1141                     THEN  PROVAR := PROVAR - 1;
000036  3  1142                     END;
000036  2  1143               IF  NEXTSYM = NIL
000037  2  1144               THEN  DECLIST[NESTLEVEL] := NIL
000044  2  1145               ELSE  BEGIN
000046  3  1146                     DECLIST[NESTLEVEL] := NEXTSYM;
000052  3  1147                     NEXTSYM^.LASTSYM := NIL;
000057  3  1148                     END;
000057  2  1149               DISPOSE (PTRVARS);
000061  2  1150               PTRVARS := DECLIST[NESTLEVEL];
000066  2  1151               END;
000067  1  1152
000067  1  1153   (* REINSERT REDEFINED KEYWORDS INTO THE SYMBOL TABLE *)
000067  1  1154   WHILE  HEADREDEF[NESTLEVEL] <> NIL  DO
000074  1  1155          WITH  HEADREDEF[NESTLEVEL]^  DO
000103  1  1156               BEGIN
000103  2  1157               PTRNEWREC := HEADREDEF[NESTLEVEL];
000106  2  1158               INSERT;
000107  2  1159               HEADREDEF[NESTLEVEL] := NEXTTMP;
000114  2  1160               END;
000115  1  1161
000115  1  1162   (* REMOVE USER DEFINED FUNCTIONS FROM THE SYMTBL *)
000115  1  1163   WHILE  HEADTEMP[NESTLEVEL] <> NIL  DO
000122  1  1164          WITH  HEADTEMP[NESTLEVEL]^  DO
000131  1  1165               BEGIN
000131  2  1166               IF  LASTSYM <> NIL
000131  2  1167               THEN  LASTSYM^.NEXTSYM := NEXTSYM
000137  2  1168               ELSE  SYMTBL[STRING[1]] := NIL;
000144  2  1169               IF  NEXTSYM <> NIL
000145  2  1170               THEN  NEXTSYM^.LASTSYM := LASTSYM;
000154  2  1171               IF  USED
000154  2  1172               THEN  UNIQUEOPS := UNIQUEOPS + 1
000156  2  1173               ELSE AMTFWDDEC := AMTFWDDEC - 1;
000163  2  1174               PTRVARS := NEXTTMP;
000165  2  1175               DISPOSE (HEADTEMP[NESTLEVEL]);
000172  2  1176               HEADTEMP[NESTLEVEL] := PTRVARS;
000177  2  1177               END;
000200  1  1178
000200  1  1179   IF  AMTFWDDEC < 0
000200  1  1180   THEN  PROCECLNOTUSED[MODULE] := TRUE;
000205  1  1181   NESTLEVEL := NESTLEVEL - 1;
000212  1  1182   1:
000212  1  1183   FWDDEC := FALSE;
000214  1  1184
000214  1  1185   END;  (* EXITPROCEDURE *)
000220  0  1186
000220  0  1187
000220  0  1188
000220  0  1189   PROCEDURE EXITNMBR;
000002  0  1190      (* COUNT AND REMOVE UNIQUE NUMBERS FROM THE NUMBER LIST *)
000002  0  1191
000002  0  1192   VAR   PTR:   SYMLINK;
000003  0  1193
000003  0  1194   BEGIN (* EXITNMBR *)
000003  1  1195
000003  1  1196   WHILE  HEADNMBR <> NIL  DO
000007  1  1197          BEGIN
```

```
000007  2        COUNT (UNOPN);
000012  2        PTR := HEADNMBR;
000014  2        HEADNMBR := PTR^.NEXTSYM;
000021  2        DISPOSE (PTR);
000023  2        END;
000024  1
000024  1  END;  (* EXITNMBR *)
000030  0
000030  0
000030  0
000030  0  FUNCTION INDENFUNC: INTEGER;
000003  0     (* COMPUTE THE INDENTING FUNCTION *)
000003  0
000003  0
000003  0  BEGIN (* INDENFUNC *)
000003  1
000003  1  INDENFUNC := ((LEFTINDENT  MOD 1000) * 1000000) +
000016  1               ((RIGHTINDENT MOD 1000) * 1000)    +
000025  1                (ZEROINDENT  MOD 1000);
000033  1
000033  1  END;  (* INDENFUNC *)
000036  0
000036  0
000036  0
000036  0  PROCEDURE INSERTPROGNOSIS;
000002  0     (* INSERT COUNTED PARAMETERS INTO PROGNOSIS *)
000002  0
000002  0  BEGIN (* INSERTPROGNOSIS *)
000002  1
000002  1  PROGNOSIS[MODULE,1]  := TOTALLINES;
000012  1  PROGNOSIS[MODULE,2]  := CODELINES;
000017  1  PROGNOSIS[MODULE,3]  := TOTALCOMMENTS;
000024  1  PROGNOSIS[MODULE,4]  := MORETHANONE;
000030  1  PROGNOSIS[MODULE,5]  := CONSANDTYPES;
000036  1  PROGNOSIS[MODULE,6]  := VARIABLES;
000043  1  PROGNOSIS[MODULE,7]  := VARNOTUSED;
000050  1  PROGNOSIS[MODULE,8]  := PROCANDFUNC;
000055  1  PROGNOSIS[MODULE,9]  := VAPPARA;
000061  1  PROGNOSIS[MODULE,10] := VALPARA;
000066  1  PROGNOSIS[MODULE,11] := PROVAR;
000073  1  PROGNOSIS[MODULE,12] := FORSTMT;
000100  1  PROGNOSIS[MODULE,13] := REPSTMT;
000105  1  PROGNOSIS[MODULE,14] := WHISTMT;
000111  1  PROGNOSIS[MODULE,15] := GO;
000116  1  PROGNOSIS[MODULE,16] := UNIQUEOPS;
000123  1  PROGNOSIS[MODULE,17] := UNIQUEOPN;
000130  1  PROGNOSIS[MODULE,18] := OPERATORS;
000135  1  PROGNOSIS[MODULE,19] := OPERANDS;
000142  1  PROGNOSIS[MODULE,20] := INDENFUNC;
000150  1
000150  1  END;  (* INSERTPROGNOSIS *)
000152  0
000152  0
000152  0
000152  0  PROCEDURE MODFINI;
000002  0     (* TERMINATE A MODULE *)
000002  0
```

```
BEGIN (* MODFINI *)

EXITNMBR;
EXITPROCEDURE;
COUNT (UNDEFS);
VARIABLES := VARIABLES - VARNOTUSED;
UNIQUEOPN  := UNIQUEOPN - VARNOTUSED;
CODELINES := CODELINES + MORETHANONE;
INSERTPROGNOSIS;

END;  (* MODFINI *)



PROCEDURE SORTPROGNOSIS (L,R:INTEGER);
   (* SORT PROGNOSIS ON THE DESIRED KEY USING A QUICKSORT. *)

VAR   PART,    (* NUMBER PARTITIONING ON *)
      LOW,     (* LEFT POINTER *)
      HIGH,    (* RIGHT POINTER *)
      TEMP:    (* TEMPORARY STORAGE *)
               INTEGER;

      (* ORDER WILL BE THE ARRAY THAT CONTAINS THE SORTED LIST *)

BEGIN (* SORTPROGNOSIS *)

      PART := ORDER[(L+R) DIV 2];
      LOW  := L;
      HIGH := R;
      REPEAT  (* PARTITION LIST *)
              WHILE  (PROGNOSIS[ORDER[LOW].KEY] < PROGNOSIS[PART.KEY]) AND (LOW < R)  DO
                     (* MOVE RIGHT *)
                     LOW := LOW + 1;
              WHILE  (PROGNOSIS[ORDER[HIGH].KEY] > PROGNOSIS[PART.KEY]) AND (HIGH > L)  DO
                     (* MOVE LEFT *)
                     HIGH := HIGH - 1;
              IF  LOW <= HIGH
              THEN  BEGIN  (* SWAP *)
                    TEMP := ORDER[LOW];
                    ORDER[LOW] := ORDER[HIGH];
                    ORDER[HIGH] := TEMP;
                    LOW := LOW + 1;
                    HIGH := HIGH - 1;
                    END;   (* SWAP *)
      UNTIL  LOW > HIGH;
      IF  L < HIGH
      THEN  SORTPROGNOSIS (L,HIGH);
      IF  LOW < R
      THEN  SORTPROGNOSIS (LOW,R);

END;  (* SORTPROGNOSIS *)



PROCEDURE HEADING  (PARAMETER: INTEGER);
   (* PRINT HEADINGS FOR OUTPUTPROGNOSIS *)
```

```
000003  0  1312
000003  0  1313  BEGIN (* HEADINGS *)
000003  1  1314
000003  1  1315  CASE  PARAMETER  OF
000005  2  1316
000005  2  1317        1:  CASE  OUTLINE  OF
000007  3  1318                  1:  WRITE ('       ');
000015  3  1319                  2:  WRITE ('TOTAL ');
000023  3  1320                  3:  WRITE ('LINES ');
000031  3  1321            END;
000040  2  1322
000040  2  1323        2:  CASE  OUTLINE  OF
000042  3  1324                  1:  WRITE ('       ');
000050  3  1325                  2:  WRITE (' CODE ');
000056  3  1326                  3:  WRITE ('LINES ');
000064  3  1327            END;
000073  2  1328
000073  2  1329        3:  CASE  OUTLINE  OF
000075  3  1330                  1:  WRITE (' CODE ');
000103  3  1331                  2:  WRITE ('COMNT ');
000111  3  1332                  3:  WRITE ('LINES ');
000117  3  1333            END;
000126  2  1334
000126  2  1335        4:  CASE  OUTLINE  OF
000130  3  1336                  1:  WRITE (' MULT ');
000136  3  1337                  2:  WRITE (' STMT ');
000144  3  1338                  3:  WRITE ('LINES ');
000152  3  1339            END;
000161  2  1340
000161  2  1341        5:  CASE  OUTLINE  OF
000163  3  1342                  1:  WRITE (' CONS ');
000171  3  1343                  2:  WRITE ('  AND ');
000177  3  1344                  3:  WRITE ('TYPES ');
000205  3  1345            END;
000214  2  1346
000214  2  1347        6:  CASE  OUTLINE  OF
000216  3  1348                  1:  WRITE ('       ');
000224  3  1349                  2:  WRITE (' DECL ');
000232  3  1350                  3:  WRITE (' VARS ');
000240  3  1351            END;
000247  2  1352
000247  2  1353        7:  CASE  OUTLINE  OF
000251  3  1354                  1:  WRITE (' VARS ');
000257  3  1355                  2:  WRITE ('  NOT ');
000265  3  1356                  3:  WRITE (' USED ');
000273  3  1357            END;
000302  2  1358
000302  2  1359        8:  CASE  OUTLINE  OF
000304  3  1360                  1:  WRITE ('PROCS ');
000312  3  1361                  2:  WRITE ('  AND ');
000320  3  1362                  3:  WRITE ('FUNCS ');
000326  3  1363            END;
000335  2  1364
000335  2  1365        9:  CASE  OUTLINE  OF
000337  3  1366                  1:  WRITE ('       ');
000345  3  1367                  2:  WRITE ('  VAR ');
000353  3  1368                  3:  WRITE ('PARAM ');
```

```
000361  3  1369         END:
000370  2  1370
000370  2  1371     10: CASE  OUTLINE  OF
000372  3  1372                 1:  WRITE ('        ');
000400  3  1373                 2:  WRITE ('  VAL  ');
000406  3  1374                 3:  WRITE ('PARAM  ');
000414  3  1375          END:
000423  2  1376
000423  2  1377     11: CASE  OUTLINE  OF
000425  3  1378                 1:  WRITE ('        ');
000433  3  1379                 2:  WRITE (' PROC  ');
000441  3  1380                 3:  WRITE (' VARS  ');
000447  3  1381         END;
000456  2  1382
000456  2  1383     12: CASE  OUTLINE  OF
000460  3  1384                 1:  WRITE ('        ');
000466  3  1385                 2:  WRITE ('  FOR  ');
000474  3  1386                 3:  WRITE ('STMTS  ');
000502  3  1387         END;
000511  2  1388
000511  2  1389     13: CASE  OUTLINE  OF
000513  3  1390                 1:  WRITE ('        ');
000521  3  1391                 2:  WRITE ('  REP  ');
000527  3  1392                 3:  WRITE ('STMTS  ');
000535  3  1393         END;
000543  2  1394
000544  2  1395     14: CASE  OUTLINE  OF
000546  3  1396                 1:  WRITE ('        ');
000554  3  1397                 2:  WRITE ('WHILE  ');
000562  3  1398                 3:  WRITE ('STMTS  ');
000570  3  1399         END;
000577  2  1400
000577  2  1401     15: CASE  OUTLINE  OF
000601  3  1402                 1:  WRITE ('        ');
000607  3  1403                 2:  WRITE (' GOTO  ');
000615  3  1404                 3:  WRITE ('STMTS  ');
000623  3  1405         END;
000632  2  1406
000632  2  1407     16: CASE  OUTLINE  OF
000634  3  1408                 1:  WRITE ('        ');
000642  3  1409                 2:  WRITE (' UNIQ  ');
000650  3  1410                 3:  WRITE ('OPERS  ');
000656  3  1411         END;
000665  2  1412
000665  2  1413     17: CASE  OUTLINE  OF
000667  3  1414                 1:  WRITE ('        ');
000675  3  1415                 2:  WRITE (' UNIQ  ');
000703  3  1416                 3:  WRITE ('OPNDS  ');
000711  3  1417         END;
000720  2  1418
000720  2  1419     18: CASE  OUTLINE  OF
000722  3  1420                 1:  WRITE ('        ');
000730  3  1421                 2:  WRITE ('TOTAL  ');
000736  3  1422                 3:  WRITE ('OPERS  ');
000744  3  1423         END;
000753  2  1424
000753  2  1425     19: CASE  OUT.INE  OF
```

```
000755  3                 1:  WRITE ('        ');
000763  3                 2:  WRITE ('TOTAL ');
000771  3                 3:  WRITE ('OPNDS ');
000777  3             END;
001006  2
001006  2       20:  CASE  OUTLINE  OF
001010  3                 1:  WRITE ('          ');
001016  3                 2:  WRITE (' INDENTING ');
001024  3                 3:  WRITE ('  FUNCTION ');
001032  3             END;
001041  2
001041  2   END:  (* PARAMETER *)
001057  1
001057  1   END:  (* HEADINGS *)
001125  0
001125  0
001125  0
001125  0   PROCEDURE OUTPUTPROGNOSIS;
000002  0      (* WRITE OUT PROGNOSIS.  KEY AND PARACOUNT ARE CONSIDERED
000002  0         SYNONOMOUS AS IT IS DESIREABLE TO HAVE THE KEY THE FAR
000002  0         RIGHT-HAND PARAMETER *)
000002  0
000002  0   VAR   I, J, PARAMETER,
000002  0         MODINDX:               INTEGER;
000006  0         STOPOUT, SAMEPAGE: BOOLEAN;
000010  0
000010  0   BEGIN (* OUTPUTPROGNOSIS *)
000010  1
000010  1   STOPOUT := FALSE;
000006  1   MODINDX := 0;
000007  1
000007  1   REPEAT
000007  2   WRITELN ('1');
000015  2   FOR OUTLINE := 1 TO 3 DO
000017  2       BEGIN
000021  3       WRITE ('      ');
000024  3       FOR PARAMETER := 1 TO PARACOUNT DO
000030  3           HEADINGS (PARAMETER);
000041  3       WRITELN;
000042  3       END;
000047  2   WRITELN;
000050  2   SAMEPAGE := TRUE;
000052  2   WHILE  SAMEPAGE AND (MODINDX < MODULE)  DO
000056  2          BEGIN
000056  3          MODINDX := MODINDX + 1;
000057  3          I := ORDER[MODINDX];
000063  3          WRITE (' ',CCID[I]);
000101  3          FOR J := 1 TO (PARACOUNT-1) DO
000104  3          WRITE (PROGNOSIS[I,J]:5,' ');
000130  3          (* PARACOUNT CONTAINS THE INDENTING FUNCTION *)
000130  3          WRITE (PROGNOSIS[I,PARACOUNT]:10);
000137  3          WRITE (' ');
000144  3       (* IF  PRODECLNOTUSED *)
000144  3          IF  PRODECLNOTUSED[I]
000147  3          THEN  WRITE('**');
000155  3       (* IF  VARNOTUSED > 0 *)
000155  3          IF  PROGNOSIS[I,7] > 0
```

```
000161  3          THEN  WRITE ('*');
000167  3          WRITELN;
000171  3          IF  ((MODINDX MOD CCIDPERPAGE) = 0)
000200  3          THEN  SAMEPAGE := FALSE;
000202  3          END;
000203  2          IF  MODINDX < MODULE
000203  2          THEN  SAMEPAGE := TRUE
000205  2          ELSE  STOPOUT := TRUE;
000207  2    UNTIL  STOPOUT;
000211  1
000211  1    WRITELN ('*    VARIABLE(S) WERE DECLARED BUT NOT USED');
000217  1    WRITELN ('**   PROCEDURE(S) WERE DECLARED BUT NOT USED');
000225  1    WRITELN ('***  VARIABLE(S) AND PROCEDURE(S) WERE DECLARED BUT NOT USED');
000233  1
000233  1    END;  (* OUTPUTPROGNOSIS *)
000273  0
000273  0
000273  0
000273  0    FUNCTION ADD (KEY1,KEY2:INTEGER): INTEGER;
000005  0      (* ADD RETURNS AS A VALUE KEY1 WHERE
000005  0         PROGNOSIS[KEY1] = PROGNOSIS[KEY1] + PROGNOSIS[KEY2] FOR ALL MODULES *)
000005  0
000005  0    VAR   I:   INTEGER;
000006  0
000006  0    BEGIN (* ADD *)
000006  1
000006  1    FOR I := 1 TO MODULE DO
000011  1        PROGNOSIS[I,KEY1] := PROGNOSIS[I,KEY1] + PROGNOSIS[I,KEY2];
000041  1    ADD := KEY1;
000043  1    ADDWASDONE := TRUE;
000044  1
000044  1    END;  (* ADD *)
000055  0
000055  0
000055  0
000055  0    PROCEDURE STATISTICS (KEY:INTEGER);
000003  0      (* DETERMINE A MATRIX OF WEIGHTS TO DETERMINE THOSE PROGRAMS THAT
000003  0         APPEAR TO BE SIMILAR *)
000003  0
000003  0    VAR   IMPORTANCE,              (* WEIGHT GIVEN TO A PARTICULAR PARAMETER *)
000003  0          DELTA,                   (* DIFFERENCE BETWEEN VALUES IN THE WINDOW *)
000003  0          BOTTOM,                  (* INDEX THAT RUNS FROM BOTTOMOFWINDOW TO TOP *)
000003  0          TOP,                     (* CURRENT TOP OF WINDOW *)
000003  0          WINDOW,                  (* CURRENT SIZE OF WINDOW *)
000003  0          BOTTOMOFWINDOW,          (* CURRENT BOTTOM OF WINDOW *)
000003  0          MINI,                    (* LEFT INDEX FOR STAT *)
000003  0          MAXI:                    (* RIGHT INDEX FOR STAT *)
000003  0                INTEGER;
000013  0
000013  0             PROCEDURE GETNEWTOP;
000002  0                (* GET A NEW TOP OF THE WINDOW *)
000002  0
000002  0             BEGIN (* GETNEWTOP *)
000002  1
000002  1             WHILE ((PROGNOSIS[ORDER[TOP].KEY] - PROGNOSIS[ORDER[BOTTOMOFWINDOW].KEY]) <= WINDOW) AND (TOP <> MODULE)
000031  1                  TOP := TOP + 1;
000033  1
```

```
000033  1              END:  (* GETNEWTOP *)
000035  0
000035  0              PROCEDURE MIN (X,Y:INTEGER);
000004  0
000004  0              BEGIN (* MIN *)
000004  1
000004  1              IF  X < Y
000004  1              THEN  BEGIN
000006  2                    MINI := X;
000007  2                    MAXI := Y;
000010  2                    END
000010  1              ELSE  BEGIN
000011  2                    MINI := Y;
000013  2                    MAXI := X;
000014  2                    END;
000014  1
000014  1              END:  (* MIN *)
000022  0
000022  0      BEGIN (* STATISTICS *)
000022  1
000022  1      CASE  KEY  OF
000005  2
000005  2           2:  BEGIN
000005  3               WINDOW := 3;
000007  3               IMPORTANCE := 5:
000010  3               END:
000011  2
000011  2           6:  BEGIN
000011  3               WINDOW := 2;
000013  3               IMPORTANCE := 3:
000014  3               END:
000015  2
000015  2          12:  BEGIN
000015  3               WINDOW := 1;
000017  3               IMPORTANCE := 2;
000020  3               END:
000021  2
000021  2          16:  BEGIN
000021  3               WINDOW := 3;
000023  3               IMPORTANCE := 5:
000024  3               END:
000025  2
000025  2          17:  BEGIN
000025  3               WINDOW := 3;
000027  3               IMPORTANCE := 5:
000030  3               END:
000031  2
000031  2          18:  BEGIN
000031  3               WINDOW := 5:
000033  3               IMPORTANCE := 6:
000034  3               END:
000035  2
000035  2          14:  BEGIN
000035  3               WINDOW := 5;
000037  3               IMPORTANCE := 8:
000040  3               END:
000041  2
```

```
000041  2        OTHERWISE  BEGIN
000058  3                   WINDOW := 1;
000060  3                   IMPORTANCE := 1;
000061  3                   END;
000061  2
000061  2        END;  (* CASE *)
000061  1
000061  1
000061  1   BOTTOMOFWINDOW := 1;
000053  1   TOP := 2;
000064  1   GETNEWTOP;
000066  1
000066  1   WHILE  BOTTOMOFWINDOW < MODULE  DO
000071  1          BEGIN
000071  2          BOTTOM := BOTTOMOFWINDOW + 1;
000072  2          WHILE  BOTTOM <= (TOP-1)  DO
000075  2                 BEGIN
000075  3                 DELTA := PROGNOSIS[ORDER[BOTTOM].KEY] - PROGNOSIS[ORDER[BOTTOMOFWINDOW].KEY];
000117  3                 MIN (ORDER[BOTTOMOFWINDOW],ORDER[BOTTOM]);
000125  3                 STAT[MINI,MAXI] := STAT[MINI,MAXI] + (IMPORTANCE-DELTA);
000140  3                 BOTTOM := BOTTOM + 1;
000141  3                 END;
000142  2          BOTTOMOFWINDOW := BOTTOMOFWINDOW + 1;
000144  2          IF  PROGNOSIS[ORDER[BOTTOMOFWINDOW].KEY] > PROGNOSIS[ORDER[BOTTOMOFWINDOW-1].KEY]
000163  2          THEN  GETNEWTOP;
000167  2          END;
000170  1
000170  1   END;  (* STATISTICS *)
000214  0
000214  0
000214  0
000214  0
000214  0   PROCEDURE MORESORTS (KEY1,KEY2,KEY3,KEY4,KEY5,KEY6,KEY7: INTEGER);
000011  0      (* MAKE ADDITIONAL SORTS AS REQUESTED.
000011  0         OUTPUT-
000011  0            LIST OF CCIDS AND SORTED LIST OF DESIGNATED KEYS *)
000011  0
000011  0  VAR
000011  0         I,J,SORTSREQUESTED:   INTEGER;
000014  0         KEYS:                 ARRAY [1..7] OF INTEGER;
000023  0         RESULTS:              ARRAY [1..7,MODULENUMBERS] OF INTEGER;
001747  0         MODINDX:              INTEGER;
001750  0         STOPOUT,SAMEPAGE:     BOOLEAN;
001752  0
001752  0   BEGIN (* MORESORTS *)
001752  1
001752  1   KEYS[1] := KEY1;
000006  1   KEYS[2] := KEY2;
000007  1   KEYS[3] := KEY3;
000010  1   KEYS[4] := KEY4;
000011  1   KEYS[5] := KEY5;
000012  1   KEYS[6] := KEY6;
000014  1   KEYS[7] := KEY7;
000015  1
000015  1   SORTSREQUESTED := 0;
000016  1
000016  1   FOR I := 1 TO 7 DO
```

```
000017  1      IF  (KEYS[I] <= PARACOUNT) AND (KEYS[I] > 0)
000030  1      THEN  BEGIN
000031  2            SORTSREQUESTED := SORTSREQUESTED + 1;
000032  2            KEY := KEYS[I];
000036  2            KEYS[SORTSREQUESTED] := KEY;
000041  2            SORTPROGNOSIS (I,MODULE);
000043  2            STATISTICS (KEY);
000045  2
000045  2            FOR J := 1 TO MODULE DO
000050  2                RESULTS[SORTSREQUESTED,J] := ORDER[J];
000065  2            END;
000072  1
000072  1  STOPOUT := FALSE;
000073  1  MODINDX := 0;
000075  1
000075  1  IF  SORTSREQUESTED > 0
000075  1  THEN  REPEAT
000077  2        WRITELN ('1');
000105  2        FOR OUTLINE := 1 TO 3 DO
000107  2            BEGIN
000111  3            FOR I := 1 TO SORTSREQUESTED DO
000114  3                BEGIN
000116  4                WRITE ('              ');
000123  4                HEADINGS (KEYS[I]);
000130  4                END;
000135  3            WRITELN;
000136  3            END;
000143  2        WRITELN;
000144  2        SAMEPAGE := TRUE;
000146  2        WHILE  SAMEPAGE AND (MODINDX < MODULE)  DO
000151  2               BEGIN
000151  3               MODINDX := MODINDX + 1;
000153  3               FOR I := 1 TO SORTSREQUESTED DO
000155  3                   BEGIN
000157  4                   WRITE ('       ',CCID[RESULTS[I,MODINDX]],' ');
000207  4                   WRITE (PROGNOSIS[RESULTS[I,MODINDX],KEYS[I]]:5,' ');
000235  4                   END;
000242  3               WRITELN;
000243  3               IF  ((MODINDX MOD CCIDPERPAGE) = 0)
000252  3               THEN  SAMEPAGE := FALSE;
000254  3               END;
000255  2        IF  MODINDX < MODULE
000255  2        THEN  SAMEPAGE := TRUE
000257  2        ELSE  STOPOUT := TRUE;
000261  2        UNTIL  STOPOUT;
000263  1
000263  1   IF  ADDWASDONE
000263  1   THEN  WRITELN ('-WARNING: HEADING(S) ARE NOT CORRECT.  AT LEAST',
000272  1                  ' ONE REFLECTS A SUM OF TWO COUNTS.');
000300  1
000300  1  END;  (* MORESORTS *)
000350  0
000350  0
000350  0
000350  0  PROCEDURE PRINTFREQ;
000002  0     (* COMPILE AND PRINT THE FREQUENCY DISTRIBUTION GRAPH FOR STAT *)
000002  0
```

```
000002  0   CONST   MINWEIGHT = 0;
000002  0           MAXWEIGHT = 32;
000002  0           MAXEXPECTED = 25;
000002  0           LOWESTCHECK = 28;
000002  0
000002  0   VAR     STILLHITS,          (* NUMBER OF 0 HIT PAIRS *)
000002  0           WEIGHT,             (* WEIGHT OF A PARTICULAR PAIR *)
000002  0           NHIT28,             (* NUMBER OF 28 HITS *)
000002  0           NHIT29,             (* NUMBER OF 29 HITS *)
000002  0           NHIT30,             (* NUMBER OF 30 HITS *)
000002  0           NHIT31,             (* NUMBER OF 31 HITS *)
000002  0           NHIT32,             (* NUMBER OF 32 HITS *)
000002  0           HINGE1,             (* FIRST HINGE FROM 5-NUMBER SUMMARY *)
000002  0           HINGE2,             (* SECOND HINGE FROM 5-NUMBER SUMMARY *)
000002  0           HINGEVAL1,          (* VALUE OF CORRELATION NUMBER AT HINGE1 *)
000002  0           HINGEVAL2,          (* VALUE OF CORRELATION NUMBER AT HINGE2 *)
000002  0           STEP,               (* DIFFERENCE IN VALUE OF TWO HINGES *)
000002  0           OUTERFENCE,         (* CORRELATIONS BEYOND THE OUTERFENCE ARE SUSPECTED OF PLAGIARISM *)
000002  0           PAIRS,              (* (MODULE * (MODULE-1)) DIV 2 *)
000002  0           MEDIAN,             (* MEDIAN OF PAIRS *)
000002  0           I,J:                (* UTILITY INDEX *)
000002  0                   INTEGER;
000023  0           LOOKING1,           (* TRUE IF HINGEVAL1 NOT FOUND *)
000023  0           LOOKING2:           (* TRUE IF HINGEVAL2 NOT FOUND *)
000023  0                   BOOLEAN;
000025  0           HITS:               (* HOLDS WEIGHT OCCURENCES FROM STAT *)
000025  0                   ARRAY [MINWEIGHT..MAXWEIGHT] OF INTEGER;
000065  0           CHECK:              (* CONTAINS PAIRS OF PROGRAMS THAT HAVE LIKE HITS *)
000066  0                   ARRAY [LOWESTCHECK..MAXWEIGHT,1..MAXEXPECTED,1..2] OF INTEGER;
000450  0
000460  0             PROCEDURE PRINTGRAPH;
000002  0                (* PRINT THE TABLE FOR THE FREQUENCY DISTRIBUTION *)
000002  0
000002  0             VAR   I:   INTEGER;
000003  0
000003  0             BEGIN (* PRINTGRAPH *)
000003  1
000003  1             WRITELN ('-');
000012  1             WRITELN (' FREQUENCY DISTRIBUTION GRAPH FOR PAIRS OF PROGRAMS');
000020  1             WRITELN ('-');
000026  1             WRITE (' ':20);
000032  1             FOR I := MINWEIGHT TO MAXWEIGHT DO
000034  1                 WRITE (I:3);
000046  1             WRITELN;
000047  1             WRITE (' ':20);
000053  1             FOR I := MINWEIGHT TO MAXWEIGHT DO
000055  1                 WRITE ('---');
000070  1             WRITELN;
000071  1
000071  1             END;  (* PRINTGRAPH *)
000104  0
000104  0   BEGIN (* PRINTFREQ *)
000104  1
000104  1   FOR I := MINWEIGHT TO MAXWEIGHT DO
000006  1       HITS[I] := 0;
000016  1   NHIT28 := 0;  NHIT29 := 0;  NHIT30 := 0;  NHIT31 := 0;  NHIT32 := 0;
000021  1   FOR I := 1 TO (MODULE-1) DO
```

```
000024  1  1768       FOR J := (I+1) TO MODULE DO
000031  1  1769          BEGIN
000033  2  1770          WEIGHT := STAT[I,J];
000042  2  1771          HITS[WEIGHT] := HITS[WEIGHT] + 1;
000047  2  1772          CASE  WEIGHT  OF
000050  3  1773               28:  BEGIN
000050  4  1774                    NHIT28 := NHIT28 + 1;
000052  4  1775                    CHECK[WEIGHT,NHIT28,1] := I;
000062  4  1776                    CHECK[WEIGHT,NHIT28,2] := J;
000071  4  1777                    END;
000072  3  1778               29:  BEGIN
000072  4  1779                    NHIT29 := NHIT29 + 1;
000074  4  1780                    CHECK[WEIGHT,NHIT29,1] := I;
000104  4  1781                    CHECK[WEIGHT,NHIT29,2] := J;
000113  4  1782                    END;
000114  3  1783               30:  BEGIN
000114  4  1784                    NHIT30 := NHIT30 + 1;
000116  4  1785                    CHECK[WEIGHT,NHIT30,1] := I;
000126  4  1786                    CHECK[WEIGHT,NHIT30,2] := J;
000135  4  1787                    END;
000136  3  1788               31:  BEGIN
000136  4  1789                    NHIT31 := NHIT31 + 1;
000140  4  1790                    CHECK[WEIGHT,NHIT31,1] := I;
000150  4  1791                    CHECK[WEIGHT,NHIT31,2] := J;
000157  4  1792                    END;
000160  3  1793               32:  BEGIN
000160  4  1794                    NHIT32 := NHIT32 + 1;
000162  4  1795                    CHECK[WEIGHT,NHIT32,1] := I;
000172  4  1796                    CHECK[WEIGHT,NHIT32,2] := J;
000201  4  1797                    END;
000202  3  1798               OTHERWISE;
000211  3  1799               END;  (* CASE *)
000211  2  1800          END;
000222  1  1801
000222  1  1802  PAIRS := (MODULE * (MODULE-1)) DIV 2;
000225  1  1803  MEDIAN := (PAIRS + 1) DIV 2;
000227  1  1804  HINGE1 := (1 + MEDIAN) DIV 2;
000231  1  1805  HINGE2 := (MEDIAN - 1) + HINGE1;
000233  1  1806  LOOKING1 := TRUE;
000234  1  1807  LOOKING2 := TRUE;
000235  1  1808  WEIGHT := 0;
000236  1  1809  I      := 0;
000237  1  1810
000237  1  1811  WHILE  (LOOKING1 OR LOOKING2)  DO
000241  1  1812         BEGIN
000241  2  1813         WEIGHT := HITS[I] + WEIGHT;
000246  2  1814         IF  LOOKING1
000246  2  1815         THEN  IF  WEIGHT >= HINGE1
000247  2  1816               THEN  BEGIN
000251  3  1817                     HINGEVAL1 := I;
000252  3  1818                     LOOKING1 := FALSE;
000253  3  1819                     END;
000253  2  1820         IF  LOOKING2
000253  2  1821         THEN  IF  WEIGHT >= HINGE2
000255  2  1822               THEN  BEGIN
000257  3  1823                     HINGEVAL2 := I;
000261  3  1824                     LOOKING2 := FALSE;
```

PASCAL COMPILER - E.T.H. ZUERICH / UNIVERSITY OF MINNESOTA. PASCAL-6000 V3.2.0. 80/12/01. 19.35.08.
UNIVERSITY OF COLORADO COMPUTING CENTER KRONOS 2.1 (80/06/23) PAGE 33

```
000262 3               END;
000262 2         I := I + 1;
000264 2         END;
000265 1 STEP := HINGEVAL2 - HINGEVAL1;
000267 1 OUTERFENCE := (3 * STEP) + HINGEVAL2;
000271 1
000271 1 WRITELN ('1');  WRITELN ('-');
000304 1 WRITELN (' TUKEY ESTIMATE FOR SUSPICION OF PLAGIARISM: ', (OUTERFENCE+1):3);
000316 1
000316 1 FOR I := MINWEIGHT TO MAXWEIGHT DO
000320 1      IF  HITS[I] > 40
000324 1      THEN  HITS[I] := 40;
000335 1 PRINTGRAPH;
000336 1 STILLHITS := 0;
000340 1 WHILE  STILLHITS < MAXWEIGHT  DO
000343 1         BEGIN
000343 2         STILLHITS := 0;
000344 2         WRITE (' ':20);
000350 2         FOR I := MINWEIGHT TO MAXWEIGHT DO
000352 2              BEGIN
000354 3              IF  HITS[I] = 0
000356 3              THEN  BEGIN
000360 4                    STILLHITS := STILLHITS + 1;
000362 4                    WRITE ('   ');
000367 4                    END
000367 3              ELSE  BEGIN
000370 4                    HITS[I] := HITS[I] - 1;
000375 4                    WRITE ('  *');
000402 4                    END;
000402 3              END;
000407 2         WRITELN;
000410 2         END;
000411 1 WRITELN ('1');
000417 1 FOR I := LOWESTCHECK TO MAXWEIGHT DO
000421 1      CASE  I  OF
000424 2             28:  IF  NHIT28 > 0  THEN
000426 2                  BEGIN
000426 3                  WRITELN ('- THE FOLLOWING PAIRS HAVE A CORRELATION OF 28:');
000434 3                  FOR J := 1 TO NHIT28 DO
000437 3                       WRITELN (' ':10,CCID[CHECK[I,J,1]],' ',CCID[CHECK[I,J,2]]);
000521 3                  END;
000522 2             29:  IF  NHIT29 > 0  THEN
000524 2                  BEGIN
000524 3                  WRITELN ('- THE FOLLOWING PAIRS HAVE A CORRELATION OF 29:');
000532 3                  FOR J := 1 TO NHIT29 DO
000535 3                       WRITELN (' ':10,CCID[CHECK[I,J,1]],' ',CCID[CHECK[I,J,2]]);
000617 3                  END;
000620 2             30:  IF  NHIT30 > 0  THEN
000622 2                  BEGIN
000622 3                  WRITELN ('- THE FOLLOWING PAIRS HAVE A CORRELATION OF 30:');
000630 3                  FOR J := 1 TO NHIT30 DO
000633 3                       WRITELN (' ':10,CCID[CHECK[I,J,1]],' ',CCID[CHECK[I,J,2]]);
000715 3                  END;
000716 2             31:  IF  NHIT31 > 0  THEN
000720 2                  BEGIN
000720 3                  WRITELN ('- THE FOLLOWING PAIRS HAVE A CORRELATION OF 31:');
000726 3                  FOR J := 1 TO NHIT31 DO
```

AD-A101 490 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH F/G 9/2
A TOOL FOR DETECTING PLAGIARISM IN PASCAL PROGRAMS.(U)
DEC 80 S L GRIER
UNCLASSIFIED AFIT-CI-80-74T NL

2 OF 2
AD A
101490

END
DATE
FILMED
8-81
DTIC

```
000731  3                           WRITELN (' ':10.CCID[CHECK[I.J.1]].'.'.CCID[CHECK[I.J.2]]);
001013  3                       END;
001014  2               32:   IF  NHIT32 > 0  THEN
001016  2                       BEGIN
001016  3                       WRITELN ('- THE FOLLOWING PAIRS HAVE A CORRELATION OF 32:');
C01024  3                       FOR J := 1 TO NHIT32 DO
001027  3                           WRITELN (' ':10.CCID[CHECK[I.J.1]].'.'.CCID[CHECK[I.J.2]]);
C0111l  3                       END;
001112  2                 END;
001126  1
001126  1   END:  (* PRINTFREQ *)
001236  0
001236  0
001236  0
001236  0   PROCEDURE COUNT;
000003  0      (* ADDS UP THE OCCURRENCES OF THE VARIOUS PARAMETERS *)
000003  0
000003  0   BEGIN (* COUNT *)
000003  1
000003  1   CASE COUNTER OF
000005  2
000005  2   EPLIN:      TOTALLINES := TOTALLINES + 1;
000010  2
000010  2   EPOPS:      OPERATORS := OPERATORS + 1;
000013  2            (* OPERATORS CAN BE DECREMENTED IN LBRAC IN DRIVER
000013  2               ASSIGNMENT OPERATORS ARE NOT COUNTED *)
000013  2
000013  2   EPCT:       CONSANDTYPES := CONSANDTYPES + 1;
000016  2
000016  2   UNOPN:      UNIQUEOPN := UNIQUEOPN + 1;
000021  2            (* VARNOTUSED WILL BE SUBTRACTED FROM UNIQUEOPN - SEE MODFINI *)
000021  2
000021  2   EPVAR:      VARIABLES := VARIABLES + 1;
000024  2            (* VARNOTUSED WILL BE SUBTRACTED FROM VARIABLES - SEE MODFINI *)
000024  2
000024  2   VARPA:      VARPARA := VARPARA + 1;
000027  2
000027  2   PRVAR:      PROVAR := PROVAR + 1;
000032  2            (* PROVAR CAN BE DECREMENTED IN EXITPROCEDURE *)
000032  2
000032  2   EPPRO:      PROCANDFUNC := PROCANDFUNC + 1;
000035  2
000035  2   EPOPN:      OPERANDS := OPERANDS + 1;
000040  2            (* DECREMENTED IN ASSIGN IN DRIVER *)
000040  2
000040  2   FORCT:      FORSTMT := FORSTMT + 1;
000043  2
000043  2   GOCNT:      GO := GO + 1;
000046  2
000046  2   REPCT:      REPSTMT := REPSTMT + 1;
000051  2
000051  2   WHICT:      WHISTMT := WHISTMT + 1;
000054  2
000054  2   EPCOM:      TOTALCOMMENTS := TOTALCOMMENTS + LINESOFCOMMENT;
000057  2
000057  2   EPMOL:      MORETHANONE := MORETHANONE + 1;
000062  2
```

```
000062  2  1939  VALPA:     VALPARA := VALPARA + 1;
000065  2  1940
000065  2  1941  EPCL:      CODELINES := CODELINES + 1;
000070  2  1942          (* DECREMENTED BY ASSIGN IN DRIVER
000070  2  1943             INCREMENTED BY MORETHANONE - SEE MODFINI *)
000070  2  1944
000070  2  1945  INDNF:     (* INDENTING FUNCTION *)
000070  2  1946             IF  (LINE[INDEX]<>'(') AND (LINE[INDEX+1]<>'*')
000100  2  1947             THEN  BEGIN
000102  3  1948                   COUNT (EPCL);
000105  3  1949                   IF  ABS (STARTPOS - NUMBLNKS) > SIGINDENT
000110  3  1950                   THEN  BEGIN
000111  4  1951                         IF  (STARTPOS - NUMBLNKS) > 0
000112  4  1952                         THEN  RIGHTINDENT := RIGHTINDENT + 1
000113  4  1953                         ELSE  LEFTINDENT  := LEFTINDENT  + 1;
000117  4  1954                         STARTPOS := NUMBLNKS;
000121  4  1955                         END
000121  3  1956                   ELSE  ZEROINDENT := ZEROINDENT + 1;
000124  3  1957                   END;
000125  2  1958
000125  2  1959  UNOPS:     COUNTUNOPS;
000127  2  1960
000127  2  1961  VARNU:     VARNOTUSED := VARNOTUSED + 1;
000132  2  1962
000132  2  1963  END:  (* CASE *)
000147  1  1964
000147  1  1965  END;  (* COUNT *)
000153  0  1966
000153  0  1967
000153  0  1968
000153  0  1969  PROCEDURE DRIVER;
000002  0  1970     (* THIS IS THE DRIVER THAT MAKES THE COUNTS AND CALLS THE SCANNER.
000002  0  1971        IT UTILIZES THE FACT THAT THE PROGRAM IS COMPILEABLE. *)
000002  0  1972
000002  0  1973  LABEL   3,4;
000002  0  1974
000002  0  1975  VAR   LPARCOUNT,
000002  0  1976        I,ITEMPEND:     INTEGER;
000005  0  1977        BUFF:           CHBUFF;
000017  0  1978        PTR,
000017  0  1979        AUXPTR:         SYMLINK;
000021  0  1980        RESKEY,
000021  0  1981        STILLLBL,
000021  0  1982        STILLTYPEORCON,
000021  0  1983        STILLVAR,
000021  0  1984        STILLVAL:       BOOLEAN;
000026  0  1985
000026  0  1986             PROCEDURE PROCESSRECORD;
000002  0  1987             FORWARD;
000002  0  1988
000002  0  1989             PROCEDURE PROCESSCASE;
000002  0  1990             FORWARD;
000002  0  1991
000002  0  1992             PROCEDURE PROCESSENUMTYPE;
000002  0  1993                (* PROCESS AN ENUMERATED TYPE DECLARATION WITHIN A TYPE DECL OR CASE
000002  0  1994                   STMT.  ANY RESERVED WORDS FOUND ARE REMOVED FROM THE SYMBOL TABLE *)
000002  0  1995
```

```
000002 0   VAR   STILLENUMTYPE:  BOOLEAN;
000003 0
000003 0   BEGIN (* PROCESSENUMTYPE *)
000003 1
000003 1   STILLENUMTYPE := TRUE;
000006 1   WHILE  STILLENUMTYPE  DO
000010 1         BEGIN
000010 2         SCANNER;
000011 2         CASE  ACTION  OF
000013 3               ABT:        ABORT;
000015 3               BLNKS:      NEWLINE := FALSE;
000017 3               COMNT,
000017 3               COMMA,
000017 3               SEMI:;
000020 3               RPAR:       STILLENUMTYPE := FALSE;
000022 3               COLON:      BEGIN (* GO TO END OF CURRENT ITEM *)
000022 4                           WHILE  (ACTION <> SEMI) AND (ACTION <> RPAR)  DO
000027 4                                  BEGIN
000027 5                                  SCANNER;
000030 5                                  CASE  ACTION  OF
000032 6                                        BLNKS:     NEWLINE := FALSE;
000034 6                                        RPAR:      STILLENUMTYPE := FALSE;
000036 6                                        OTHERWISE:
000050 6                                        END;
000050 5                                  END;
000051 4                           END;
000052 3               OTHERWISE   BEGIN
000075 4                           IF  KEYWORD
000075 4                           THEN  IF  ISADELIMITER
000077 4                                 THEN  ABORT
000100 4                                 ELSE  REDEF;
000103 4                           COUNT (UNOPN);
000106 4                           COUNT (EPVAR);
000111 4                           DECLARE;
000112 4                           END;
000112 3               END;
000112 2
000112 2         END;
000113 1   END;  (* PROCESSENUMTYPE *)
000117 0
000117 0   PROCEDURE PROCESSRECORD;
000002 0      (* PROCESS A RECORD WITHIN A DECL *)
000002 0
000002 0   VAR   LHS,
000002 0         STILLRECORD:   BOOLEAN;
000004 0
000004 0   BEGIN (* PROCESSRECORD *)
000004 1
000004 1   LHS := TRUE;
000006 1   STILLRECORD := TRUE;
000007 1   WHILE  STILLRECORD  DO
000011 1         BEGIN
000011 2         IF  YES
000011 2         THEN  SCANNER;
000013 2         CASE  ACTION  OF
000015 3               ABT:           ABORT;
000017 3               BLNKS:         NEWLINE := FALSE;
```

```
000021  3  2053                      COMMA,
000021  3  2054                      COMMT::
000022  3  2055                      SEMI:          LHS := TRUE;
000024  3  2056                      COLON:         LHS := FALSE;
000026  3  2057                      LPAR:          PROCESSENUMTYPE;
000031  3  2058                      RECDEC:        PROCESSRECORD;
000034  3  2059                      CASEST:        PROCESSCASE;
000037  3  2060                      FINDEND:       STILLRECORD := FALSE;
000041  3  2061                      OTHERWISE      IF  LHS
000068  3  2062                                     THEN  BEGIN
000070  4  2063                                             IF  KEYWORD
000070  4  2064                                             THEN  IF  ISADELIMITER
000071  4  2065                                                   THEN  ABORT
000073  4  2066                                                   ELSE  REDEF;
000076  4  2067                                             COUNT (UNOPN);
000101  4  2068                                             COUNT (EPVAR);
000104  4  2069                                             DECLARE;
000105  4  2070                                             END;
000105  3  2071                      END;
000105  2  2072                 END;
000106  1  2073
000106  1  2074           END;  (* PROCESSRECORD *)
000114  0  2075
000114  0  2076           PROCEDURE PROCESSCASE;
000002  0  2077              (* PROCESS A CASE STMT WITHIN A DECL *)
000002  0  2078
000002  0  2079           VAR   STILLCASE:   BOOLEAN;
000003  0  2080
000003  0  2081                      PROCEDURE TAKEITOUT;
000002  0  2082                         (* STORE THE CONTENTS OF IDENBUFF *)
000002  0  2083
000002  0  2084                      VAR   I:   INTEGER;
000003  0  2085
000003  0  2086                      BEGIN (* TAKEITOUT *)
000003  1  2087
000003  1  2088                      FOR I := 1 TO ENDIDEN DO
000007  1  2089                          BUFF[I] := IDENBUFF[I];
000024  1  2090                      TEMPEND := ENDIDEN;
000026  1  2091
000026  1  2092                      END;  (* TAKEITOUT *)
000032  0  2093
000032  0  2094                      PROCEDURE PUTITBACK;
000002  0  2095                         (* REINSERT THE CONTENTS OF IDENBUFF *)
000002  0  2096
000002  0  2097                      VAR   I:   INTEGER;
000003  0  2098
000003  0  2099                      BEGIN (* PUTITBACK *)
000003  1  2100
000003  1  2101                      FOR I := 1 TO TEMPEND DO
000007  1  2102                          IDENBUFF[I] := BUFF[I];
000025  1  2103                      ENDIDEN := TEMPEND;
000031  1  2104
000031  1  2105                      END;  (* PUTITBACK *)
000035  0  2106
000035  0  2107           BEGIN (* PROCESSCASE *)
000035  1  2108
000035  1  2109           SCANNER;
```

PASCAL COMPILER - E.T.H. ZUERICH / UNIVERSITY OF MINNESOTA.
UNIVERSITY OF COLORADO COMPUTING CENTER

PASCAL-6000 V3.2.0. 80/12/01. 10.35 OR.
KRONOS 2.1 (80/06/23) PAGE 38

```
000005  1   WHILE  ACTION = BLNKS  DO
000010  1          BEGIN  (* SKIP BLANKS *)
000010  2          NEWLINE := FALSE;
000011  2          SCANNER;
000012  2          END;
000013  1   RESKEY := FALSE;
000015  1   IF  KEYWORD
000015  1   THEN  BEGIN  (* SAVE CURRENT ITEM *)
000016  2         PTR := PTRSYMTBL;
000020  2         RESKEY := TRUE;
000021  2         END;
000021  1   TAKEITOUT;
000023  1   SCANNER;
000024  1   WHILE  ACTION = BLNKS  DO
000027  1          BEGIN  (* SKIP BLANKS *)
000027  2          NEWLINE := FALSE;
000030  2          SCANNER;
000031  2          END;
000032  1   IF  ACTION = COLON  (* CHECK IF ITEM WAS A TAG *)
000032  1   THEN  BEGIN
000034  2         PUTITBACK;
000036  2         COUNT (UNOPN);
000041  2         COUNT (EPVAR);
000044  2         IF  RESKEY
000044  2         THEN  BEGIN  (* REDEFINE KEYWORD *)
000048  3               PTRSYMTBL := PTR;
000050  3               REDEF;
000051  3               END;
000051  2         DECLARE;
000052  2         END;
000052  1   WHILE  ACTION <> FINDOF  DO
000055  1          BEGIN
000055  2          SCANNER;
000056  2          CASE  ACTION  OF
000060  3                ABT:        ABORT;
000062  3                BLNKS:      NEWLINE := FALSE;
000064  3                LPAR:       BEGIN
000064  4                            COUNT (EPCT);
00[illegible]  4                            PROCESSENUMTYPE;
00[illegible]  4                            END;
00[illegible]  3                OTHERWISE;
00[illegible]  3                END;
00[illegible]  2          END;
00[illegible]  1   STILLCASE := TRUE;
00[illegible]  1   WHILE  STILLCASE  DO
000116  1          BEGIN
000116  2          IF  YES
000116  2          THEN  SCANNER;
00012[illegible]  2          CASE  ACTION  OF
0001[illegible]  3                ABT:        ABORT;
0001[illegible]  3                BLNKS:      NEWLINE := FALSE;
0001[illegible]  3                COMNT,
0001[illegible]  3                COMMA,
0001[illegible]  3                SEMI,
0001[illegible]  3                COLON::
0001[illegible]  3                LPAR:       PROCESSENUMTYPE;
0001[illegible]  3                RECDEC:     PROCESSRECORD;
```

```
000135  3                              CASEST:    PROCESSCASE;
000140  3                              FINDEND:   BEGIN
000140  4                                         STILLCASE := FALSE;
000142  4                                         YES := FALSE;
000143  4                                         END;
000144  3                              OTHERWISE;
000171  3                              END;
000171  2                        END;
000172  1
000172  1              END;  (* PROCESSCASE *)
000176  0
000176  0  BEGIN (* DRIVER *)
000176  1
000176  1  WHILE  TRUE  DO
000006  1  BEGIN (* WHILE *)
000006  2
000006  2  IF  YES  (* SCAN IF REQUIRED *)
000006  2  THEN  SCANNER;
000010  2  YES := TRUE;
000012  2  IF  KEYWORD
000012  2  THEN  IF  PTRSYMTBL^.OP
000020  2        THEN  BEGIN  (* COUNT OPERATORS *)
000021  3              PTRSYMTBL^.USED := TRUE;
000025  3              COUNT (EPOPS);
000030  3              END;
000030  2
000030  2  CASE  ACTION  OF  (* PERFORM THE REQUESTED ACTION *)
000032  3
000032  3  LBLDEC:    (* IGNORE ALL OF THE LABEL DECLARATION *)
000032  3             BEGIN
000032  4             STILLLBL := TRUE;
000034  4             WHILE  STILLLBL  DO
000036  4                    BEGIN  (* LOOK FOR THE END OF THE LABEL DECL *)
000036  5                    SCANNER;
000037  5                    CASE  ACTION  OF
000041  6                          PROCS,
000041  6                          LBLDEC,CONDEC,
000041  6                          TYPDEC,VARDEC,
000041  6                          VALDEC,FINDBEG:  BEGIN
000041  7                                           STILLLBL := FALSE;
000043  7                                           YES := FALSE;
000044  7                                           END;
000045  6                          BLNKS:           NEWLINE := FALSE;
000047  6                          OTHERWISE;
000067  6                          END;
000067  5                    END;
000070  4             END;
000071  3
000071  3  CONDEC,
000071  3  TYPDEC:    (* COUNT THE NUMBER OF TYPE DECLARATIONS AND CONSTANTS *)
000071  3             BEGIN
000071  4             STILLTYPEORCON := TRUE;
000073  4             WHILE  STILLTYPEORCON  DO
000075  4                    BEGIN  (* LOOK FOR THE END OF THE TYPE OR CONST DECL *)
000075  5                    SCANNER;
000076  5                    CASE  ACTION  OF
000100  6                          PROCS,
```

```
000100  6                     LBLDEC.CONDEC.
000100  6                     TYPDEC.VARDEC.
000100  6                     VALDEC.FINDBEG: BEGIN
000100  7                                     STILLTYPEORCON := FALSE;
000102  7                                     YES := FALSE;
000103  7                                     END;
000104  6                     EQUAL:          COUNT (EPCT);
000110  6                     CASEST:         PROCESSCASE;
000113  6                     RECDEC:         PROCESSRECORD;
000118  6                     LPAR:           PROCESSENUMTYPE;
000121  6                     BLNKS:          NEWLINE := FALSE;
000123  6                     OTHERWISE:
000152  6                     END;
000152  5               END;
000153  4           END;
000154  3
000154  3   VARDEC:  (* COUNT THE VARIABLES AND ENTER THEM ON TO THE LIST OF
000154  3                DECLARED VARIABLES *)
000154  3            BEGIN
000154  4            STILLVAR := TRUE;
000156  4
000156  4            IF  NOT PARADEC  THEN  (* IF NOT WITHIN A PARAMETER DECL *)
000160  4
000160  4            WHILE  STILLVAR  DO
000162  4                 BEGIN
000162  5                 SCANNER;
000163  5                 CASE  ACTION  OF
000165  6                     PROCS,
000165  6                     LBLDEC.CONDEC.
000165  6                     TYPDEC.VARDEC.
000165  6                     VALDEC.FINDBEG: BEGIN  (* FIND THE END OF THE DECL *)
000165  7                                     STILLVAR := FALSE;
000167  7                                     YES := FALSE;
000170  7                                     END;
000171  6                     COLON:          WHILE  ACTION <> SEMI  DO
000174  6                                            BEGIN  (* SKIP TO THE SEMICOLON *)
000174  7                                            SCANNER;
000175  7                                            CASE  ACTION  OF
000177  8                                                  BLNKS:     NEWLINE := FALSE;
000201  8                                                  LPAR:      PROCESSENUMTYPE;
000204  8                                                  RECDEC:    PROCESSRECORD;
000207  8                                                  CASEST:    PROCESSCASE;
000212  8                                                  OTHERWISE;
000234  8                                                  END;
000234  7                                            END;
000236  6                     BLNKS:          NEWLINE := FALSE;
000240  6                     SEMI,
000240  6                     COMNT,
000240  6                     COMMA::
000241  6                     ABT:            ABORT;
000243  6                     OTHERWISE       BEGIN  (* FOUND A VARIABLE NAME *)
000273  7                                     COUNT (UNOPN);
000270  7                                     COUNT (EPVAR);
000301  7                                     IF  KEYWORD
000301  7                                     THEN  IF  ISADELIMITER
000303  7                                           THEN  ABORT
000304  7                                           ELSE  REDEF;
```

```
000307  7  2281                                          DECLARE:
000310  7  2282                                          END:
000310  6  2283                         END;
000310  5  2284                    END
000310  4  2285
000310  4  2286           ELSE  (* IF WITHIN A PARAMETER DECL *)
000312  4  2287
000312  4  2288           WHILE  STILLVAR  DO
000314  4  2289                  BEGIN
000314  5  2290                  SCANNER;
000315  5  2291                  CASE  ACTION  OF
000317  6  2292                        ABT:         ABORT;
000321  6  2293                        BLNKS:       NEWLINE := FALSE;
000323  6  2294                        COMNT:;
000324  6  2295                        SEMI:        STILLVAR := FALSE;  (* REACHED END OF CURRENT ITEM *)
000326  6  2296                        PROCS:       ABORT;
000330  6  2297                        COMMA:;
000331  6  2298                        COLON:       BEGIN  (* GO TO THE END OF THE CURRENT ITEM *)
000331  7  2299                                     WHILE  (ACTION <> SEMI) AND (ACTION <> RPAR)  DO
000336  7  2300                                            BEGIN
000336  8  2301                                            SCANNER;
000337  8  2302                                            CASE  ACTION  OF
000341  9  2303                                                  BLNKS:       NEWLINE := FALSE;
000343  9  2304                                                  LPAR:        PROCESSENUMTYPE;
000346  9  2305                                                  RECDEC:      PROCESSRECORD;
000351  9  2306                                                  CASEST:      PROCESSCASE;
000354  9  2307                                                  RPAR:        BEGIN  (* END OF THE PARAMETER DECL *)
000354 10  2308                                                               STILLVAR := FALSE;
000356 10  2309                                                               YES := FALSE;
000357 10  2310                                                               END;
000360  9  2311                                                  OTHERWISE:
000402  9  2312                                                  END;
000402  8  2313                                            END;
000403  7  2314                                     WHILE  STILLVAR  DO
000405  7  2315                                            BEGIN  (* GO TO THE NEXT ITEM *)
000405  8  2316                                            SCANNER;
000406  8  2317                                            CASE  ACTION  OF
000410  9  2318                                                  SEMI::
000411  9  2319                                                  COMNT::
000412  9  2320                                                  BLNKS:       NEWLINE := FALSE;
000414  9  2321                                                  OTHERWISE    BEGIN
000423 10  2322                                                               STILLVAR := FALSE;
000425 10  2323                                                               YES := FALSE;
000428 10  2324                                                               CASE  ACTION  OF
000430 11  2325                                                                     VARDEC.  (* NEXT VAR PARAMETER *)
000430 11  2326                                                                     RPAR::   (* END OF PARAMETER DECL *)
000431 11  2327                                                                     PROCS:      BEGIN  (* PROCEDURE PAR/METER *)
000431 12  2328                                                                                 COUNT (VALPA);
000434 12  2329                                                                                 END;
000435 11  2330                                                                     OTHERWISE   ACTION := OPND;  (* FOUND VAR PARA
000451 11  2331                                                                     END;
000451 10  2332                                                               END;
000451  9  2333                                                  END;
000451  8  2334                                            END;
000452  7  2335                                     END;
000453  6  2336                        OTHERWISE    BEGIN  (* FOUND VAR PARAMETER *)
000501  7  2337                                     IF  KEYWORD
```

```
000501  7                                  THEN  IF  ISADELIMITER
000503  7                                            THEN  ABORT
000504  7                                            ELSE  REDEF;
000507  7                                  DECLARE;
000510  7                                  COUNT (VARPA);
000513  7                                  COUNT (EPVAR);
000516  7                                  COUNT (UNOPN);
000521  7                                  END;
000521  6                            END;
000521  5                      END;
000522  4                END;
000523  3
000523  3   VALDEC:      (* IGNORE THE VALUE DECLARATION *)
000523  3                BEGIN
000523  4                STILLVAL := TRUE;
000525  4                WHILE  STILLVAL  DO
000527  4                      BEGIN  (* LOOK FOR THE END OF THE VALUE DECL *)
000527  5                      SCANNER;
000530  5                      CASE  ACTION  OF
000532  6                            PROCS,
000532  6                            LBLDEC,CONDEC,
000532  6                            TYPDEC,VARDEC,
000532  6                            VALDEC,FINDBEG: BEGIN
000532  7                                            STILLVAL := FALSE;
000534  7                                            YES := FALSE;
000535  7                                            END;
000536  6                            BLNKS:          NEWLINE := FALSE;
000540  6                            OTHERWISE;
000560  6                            END;
000560  5                      END;
000561  4                END;
000562  3
000562  3   PROC:        (* ENTER THE CCID INTO THE CCID ARRAY
000562  3                   AND INITIALIZE STARTPOS *)
000562  3                BEGIN
000562  4                STARTPOS := 1;
000564  4                WHILE  LINE[STARTPOS] = ' '  DO  (* FIND THE INDENFUNC REFERENCE *)
000571  4                      STARTPOS := STARTPOS + 1;
000573  4                NESTLEVEL := 1;
000575  4                (* INITIALIZE LISTS *)
000575  4                HEADTEMP[NESTLEVEL] :=  NIL;
000600  4                HEADREDEF[NESTLEVEL] := NIL;
000603  4                DECLIST[NESTLEVEL] :=   NIL;
000606  4                WHILE  ACTION <> OPND  DO
000611  4                      BEGIN  (* GET THE CCID *)
000611  5                      SCANNER;
000612  5                      CASE  ACTION  OF
000614  6                            BLNKS:       NEWLINE := FALSE;
000616  6                            OTHERWISE;
000623  6                            END;
000623  5                      END;
000624  4                FOR I := 1 TO CCIDLNGTH DO  (* THE CCID OCCUPIES THE FIRST FOUR CHARS *)
000628  4                    CCID[MODULE,I] := IDENBUFF[I];
000653  4                END;
000654  3
000654  3   PROCS:       (* PROCESS A PROCEDURE DECLARATION AND ADD PARAMETERS TO THE
000654  3                   DECLIST OF THE PROPER NESTLEVEL *)
```

```
000654  3          BEGIN
000654  4          IF  NOT PRODECL  (* IF NOT A PARAMETER *)
000654  4          THEN  BEGIN
000656  5                SCANNER;
000657  5                WHILE  ACTION = BLNKS  DO
000662  5                       BEGIN  (* SKIP BLANKS *)
000662  6                       NEWLINE := FALSE;
000663  6                       SCANNER;
000664  6                       END;
000665  5                IF  KEYWORD
000665  5                THEN  IF  ISADELIMITER
000667  5                      THEN  ABORT
000670  5                      ELSE  REDEF;
000673  5                NEW (PTRNEWREC);
000675  5                WITH  PTRNEWREC^  DO
000702  5                      BEGIN  (* INSERT PROCEDURE NAME INTO SYMTBL *)
000702  6                      FOR I := 1 TO ENDIDEN DO
000705  6                          STRING[I] := IDENBUFF[I];
000725  6                      FOR I := ENDIDEN + 1 TO 10 DO
000727  6                          STRING[I] := ' ';
000743  6                      LNGTH := ENDIDEN;
000745  6                      OP := TRUE;
000746  6                      USED := FALSE;
000747  6                      NEXTSYM := NIL;
000750  6                      LASTSYM := NIL;
000752  6                      NEXTTMP := NIL;
000753  6                      TOK := FUNCS;
000755  6                      END;
000755  5                INSERTTEMP (NESTLEVEL);
000757  5                NESTLEVEL := NESTLEVEL + 1;
000763  5                (* INITIALIZE LISTS *)
000763  5                DECLIST[NESTLEVEL] :=   NIL;
000766  5                HEADTEMP[NESTLEVEL] :=  NIL;
000770  5                HEADREDEF[NESTLEVEL] := NIL;
000773  5                STARTPROC[NESTLEVEL] := NUMEEG;
000776  5                PRODECL := TRUE;
001000  5                COUNT (EPPRO);
001003  5                END
001003  4          ELSE  BEGIN  (* IF THE PROCEDURE IS A PARAMETER *)
001004  5                SCANNER;
001005  5                WHILE  ACTION = BLNKS  DO
001010  5                       BEGIN  (* SKIP BLANKS *)
001010  6                       NEWLINE := FALSE;
001011  6                       SCANNER;
001012  6                       END;
001013  5                IF  KEYWORD
001013  5                THEN  IF  ISADELIMITER
001015  5                      THEN  ABORT
001016  5                      ELSE  REDEF;
001021  5                NEW (PTRNEWREC);
001023  5                WITH  PTRNEWREC^  DO
001030  5                      BEGIN  (* INSERT PROCEDURE NAME INTO SYMTBL FOR LIFE OF PROCEDURE *)
001030  6                      FOR I := 1 TO ENDIDEN DO
001033  6                          STRING[I] := IDENBUFF[I];
001053  6                      LNGTH := ENDIDEN;
001055  6                      OP := TRUE;
001056  6                      USED := FALSE;
```

```
001057  6                      NEXTSYM := NIL;
001060  6                      LASTSYM := NIL;
001062  6                      NEXTIMP := NIL;
001063  6                      TOK := FUNCS;
001065  6                      END;
001065  5                 INSERTTEMP (NESTLEVEL);
001067  5                 STILLVAR := TRUE;
001071  5                 WHILE  STILLVAR  DO
001073  5                      BEGIN
001073  6                      SCANNER;
001074  6                      CASE  ACTION  OF
001076  7                           SEMI,
001076  7                           COMMT:,
001077  7                           COLON:      WHILE  (ACTION <> SEMI) AND (ACTION <> RPAR)  DO
001104  7                                            BEGIN  (* GO TO END OF CURRENT ITEM *)
001104  8                                            SCANNER;
001105  8                                            CASE  ACTION  OF
001107  9                                                 BLNKS:        NEWLINE := FALSE;
001111  9                                                 RPAR:         BEGIN  (* END OF PARAMETER DECL *)
001111 10                                                               STILLVAR := FALSE;
001113 10                                                               YES := FALSE;
001114 10                                                               END;
001115  9                                                 OTHERWISE:
001127  9                                                 END;
001127  8                                            END;
001131  7                           BLNKS:      NEWLINE := FALSE;
001133  7                           LPAR:       BEGIN  (* SKIP TO END OF NESTED PARAMETER DECL *)
001133  8                                       LPARCOUNT := 1;
001135  8                                       WHILE  (LPARCOUNT > 0  DO
001137  8                                            BEGIN
001137  9                                            SCANNER;
001140  9                                            CASE  ACTION  OF
001142 10                                                 BLNKS:     NEWLINE := FALSE;
001144 10                                                 LPAR:      LPARCOUNT := LPARCOUNT + 1;
001147 10                                                 RPAR:      LPARCOUNT := LPARCOUNT - 1;
001152 10                                                 OTHERWISE:
001165 10                                                 END;
001155  9                                            END;
001166  8                                       END;
001167  7                           OTHERWISE   BEGIN  (* FIND BEGINNING OF NEXT ITEM *)
001211  8                                       STILLVAR := FALSE;
001213  8                                       YES := FALSE;
001214  8                                       CASE  ACTION  OF
001216  9                                           VARDEC.   (* FOUND A VAR PARAMETER *)
001216  9                                           RPAR::    (* END OF THE PARAMETER DECL *)
001217  9                                           PROCS:        BEGIN  (* FOUND A PROCEDURE PARAMETER *)
001217 10                                                         COUNT (VALPA);
001222 10                                                         END;
001223  9                                           OTHERWISE     ACTION := OPND;  (* FOUND A VALUE PARAMETER *)
001237  9                                           END;
001237  8                                       END;
001237  7                           END;
001237  6                      END;
001240  5                 END;
001240  4            END;
001241  3
001241  3    FWD:       (* TERMINATE PROCEDURE DECL *)
```

```
001241  3  2509           BEGIN
001241  4  2510           FWDDEC := TRUE;
001243  4  2511           AMTFWDDEC := AMTFWDDEC + 1;
001245  4  2512           PROCANDFUNC := PROCANDFUNC - 1;
001246  4  2513           EXITPROCEDURE;
001247  4  2514           END;
001250  3  2515
001250  3  2516  CASEST:  (* INDICATE TO FINDEND THERE IS A CASE STMT.
001250  3  2517              REMOVE THE ARGUMENT FROM THE DECLIST  *)
001250  3  2518           BEGIN
001250  4  2519           NUMCASE := NUMCASE + 1;
001252  4  2520           SCANNER;
001253  4  2521           WHILE  ACTION = BLNKS  DO
001256  4  2522                  BEGIN  (* SKIP BLANKS *)
001256  5  2523                  NEWLINE := FALSE;
001257  5  2524                  SCANNER;
001260  5  2525                  END;
001261  4  2526           OPNUSED;
001262  4  2527           COUNT (EPOPN);
001265  4  2528           END;
001266  3  2529
001266  3  2530  FINDBEG:  (* INCREMENT NUMBER OF BEGINS *)
001266  3  2531           NUMBEG := NUMBEG + 1;
001271  3  2532
001271  3  2533  FINDEND:  (* DECREMENT NUMBER OF BEGINS; DECREMENT NESTLEVEL IF AT THE
001271  3  2534              END OF A PROCEDURE *)
001271  3  2535           BEGIN
001271  4  2536           IF  NUMCASE = 0
001271  4  2537           THEN  NUMBEG := NUMBEG - 1
001273  4  2538           ELSE  NUMCASE := NUMCASE - 1;
001277  4  2539           IF  STARTPROC[NESTLEVEL] = NUMBEG  (* TERMINATE PROCEDURE IF AT END *)
001302  4  2540           THEN  EXITPROCEDURE;
001305  4  2541           ENDFLAG := TRUE;
001307  4  2542           END;
001310  3  2543
001310  3  2544  LPAR:     (* IF IN A PROCEDURE DECL NOTIFY DRIVER *)
001310  3  2545           IF  PRODECL
001310  3  2546           THEN  BEGIN  (* IN A PROCEDURE DECLARATION *)
001312  4  2547                 PARADEC := TRUE;
001313  4  2548                 SCANNER;
001314  4  2549                 WHILE  ACTION = BLNKS  DO
001317  4  2550                        BEGIN  (* SKIP BLANKS *)
001317  5  2551                        NEWLINE := FALSE;
001320  5  2552                        SCANNER;
001321  5  2553                        END;
001322  4  2554                 CASE  ACTION  OF
001323  5  2555                       VARDEC,   (* FOUND VAR PARAMETER *)
001323  5  2556                       SEMI::    (* FOUND NO PARAMETER *)
001324  5  2557                       RPAR,     (* FOUND END OF DECL *)
001324  5  2558                       COMNT,
001324  5  2559                       PROCS:          BEGIN  (* FOUND PROCEDURE PARAMETER *)
001324  6  2560                                       COUNT (VALPA);
001330  6  2561                                       END;
001331  5  2562                       OTHERWISE       ACTION := OPND;  (* FOUND VAL PARAMETER *)
001351  5  2563                       END;
001351  4  2564                 YES := FALSE;  (* DO NOT SCAN AGAIN- LET DRIVER TAKE APPROPRIATE ACTION *)
001353  4  2565                 END;
```

```
001354  3
001354  3   RPAR:     (* IF IN A PROCEDURE, NOTIFY DRIVER OF ITS TERMINATION *)
001354  3             IF  PRODECL
001354  3             THEN  PARADEC := FALSE;
001360  3
001360  3   FORST:    (* INCREMENT FOR STMT COUNT *)
001360  3             COUNT (FORCT);
001364  3
001364  3   GOTOST:   (* INCREMENT GOTO STMT COUNT *)
001364  3             COUNT (GOCNT);
001370  3
001370  3   INST:     (* NOTIFY DRIVER THAT YOU ARE IN AN IN STMT *)
001370  3             ININ := TRUE;
001372  3
001372  3   REPST:    (* INCREMENT REPEAT STMT COUNT *)
001372  3             COUNT (REPCT);
001376  3
001376  3   WHIST:    (* INCREMENT WHILE STMT COUNT *)
001376  3             COUNT (WHICT);
001402  3
001402  3   COMNT:    (* INCREMENT LINESOFCOMMENT *)
001402  3             IF  NOT PRODECL
001402  3             THEN  COUNT (EPCOM);
001411  3
001411  3   LBRAC:    (* IF IN AN IN STMT, LBRAC IS NOT COUNTED AS AN OPERATOR *)
001411  3             BEGIN
001411  4             IF  ININ
001411  4             THEN  OPERATORS := OPERATORS - 1;
001415  4             ININ := FALSE;
001417  4             END;
001420  3
001420  3   SEMI:     (* TERMINATE PROCEDURE DECLARATIONS AND CHECK TO SEE IF MORE
001420  3                THAN ONE STMT IS ON THE SAME LINE *)
001420  3             BEGIN
001420  4             IF  (NOT PARADEC) AND PRODECL
001422  4             THEN  PRODECL := FALSE;
001424  4             IF  SAMELINE
001424  4             THEN  COUNT (EPMOL);
001431  4             IF  NOT PARADEC
001431  4             THEN  SAMELINE := TRUE;
001435  4             ENDFLAG := FALSE;
00143?  ?             END;
001440  3
001440  3   OPND:     (* INCREMENT THE APPROPRIATE OPERAND RELATED COUNTS AND ADD
0014??  3                APPROPRIATE VARIABLES TO THE DECLIST *)
00144?  3             BEGIN
001440  4             IF  PARADEC
001440  4             THEN  BEGIN  (* IF WITHIN A PARAMETER DECL *)
0014??  5                   IF  KEYWORD
0014??  5                   THEN  IF  ISADELIMITER
0014??  5                         THEN  ABORT
0014??  5                         ELSE  REDEF;
0014??  5                   COUNT (VALPA);
0014??  5                   COUNT (EPVAR);
0014??  5                   COUNT (UNOPN);
0014??  5                   DECL...E;
0014??  5                   S...LVAL := TRU...
```

```
001484  5  2623          WHILE  STILLVAL  DO
001486  5  2624                 BEGIN  (* FIND NEXT ITEM *)
001486  6  2625                 SCANNER;
001467  6  2626                 CASE  ACTION  OF
001471  7  2627                        BLNKS:     NEWLINE := FALSE;
001473  7  2628                        COMNT,
001473  7  2629                        COMMA::
001474  7  2630                        PROCS:     BEGIN  (* FOUND PROCEDURE PARAMETER *)
001474  8  2631                                   COUNT (UNOPS);
001477  8  2632                                   COUNT (EPOPS);
001502  8  2633                                   COUNT (VALPA);
001505  8  2634                                   STILLVAL := FALSE;
001507  8  2635                                   END;
001510  7  2636                        COLON:     BEGIN
001510  8  2637                                   WHILE  (ACTION <> SEMI) AND (ACTION <> RPAR)  DO
001515  8  2638                                          BEGIN  (* GO TO END OF CURRENT ITEM *)
001515  9  2639                                          SCANNER;
001516  9  2640                                          CASE  ACTION  OF
001520 10  2641                                                BLNKS:     NEWLINE := FALSE;
001522 10  2642                                                RPAR:      BEGIN  (* END OF PARAMETER DECL *)
001522 11  2643                                                           STILLVAL := FALSE;
001524 11  2644                                                           YES := FALSE;
001525 11  2645                                                           END;
001526 10  2646                                                CASEST:    PROCESSCASE;
001531 10  2647                                                OTHERWISE;
001546 10  2648                                                END;
001546  9  2649                                          END;
001547  8  2650                                   WHILE  STILLVAL  DO
001551  8  2651                                          BEGIN  (* GET NEXT ITEM *)
001551  9  2652                                          SCANNER;
001552  9  2653                                          CASE  ACTION  OF
001554 10  2654                                                SEMI,
001554 10  2655                                                COMNT::
001555 10  2656                                                BLNKS:    NEWLINE := FALSE;
001557 10  2657                                                OTHERWISE  BEGIN
001566 11  2658                                                          STILLVAL := FALSE;
001570 11  2659                                                          YES := FALSE;
001571 11  2660                                                          CASE  ACTION  OF
001573 12  2661                                                                VARDEC.  (* FIND VAR PARAMETER *)
001573 12  2662                                                                RPAR::   (* END OF PARAMETER DECL *)
001574 12  2663                                                                PROCS:      BEGIN  (* FIND PROCEDURE PARA *)
001574 13  2664                                                                            COUNT (VALPA);
001577 13  2665                                                                            END;
001600 12  2666                                                                OTHERWISE  ACTION := OPND;  (* FIND VAL PARA
001614 12  2667                                                                END;
001614 11  2668                                                          END;
001614 10  2669                                                END;
001614  9  2670                                          END;
001615  8  2671                                   END;
001616  7  2672                        OTHERWISE  BEGIN  (* FOUND ANOTHER VAL PARAMETER *)
001644  8  2673                                   IF  KEYWORD
001644  8  2674                                   THEN  IF  ISADELIMITER
001646  8  2675                                         THEN  ABORT
001647  8  2676                                         ELSE  REDEF;
001652  8  2677                                   DECLARE;
001653  8  2678                                   COUNT (VALPA);
001656  8  2679                                   COUNT (EPVAR);
```

```
001661  8                                   COUNT (UNOPN);
001664  8                                   END;
001664  7                              END;
001664  6                         END;
001665  5               END
001665  4       ELSE    BEGIN  (* NOT WITHIN A PARAMETER DECL *)
001666  5               COUNT (EPOPN);
001671  5               OPNUSED;
001672  5               END;
001672  4       END;
001673  3
001673  3  NMBR:  (* ADD NUMBER TO THE UNIQUE NUMBER LIST AND INCREMENT OPERAND COUNT *)
001673  3       BEGIN
001673  4       COUNT (EPOPN);
001676  4       NEW (PTRNEWREC);  (* INSERT NUMBERS INTO THE NUMBER LIST *)
001700  4       IF  HEADNMBR = NIL
001700  4       THEN  HEADNMBR := PTRNEWREC
001703  4       ELSE  BEGIN
001705  5               PTR := HEADNMBR;
001707  5               WHILE  PTR <> NIL  DO
001712  5                      BEGIN  (* IS IT IN THE NUMBER LIST ALREADY *)
001712  6                      IF  ENDIDEN = PTR^.LNGTH
001716  6                      THEN  WITH PTR^  DO
001724  6                             BEGIN
001724  7                             UNPACK (STRING,BUFF,1);
001730  7                             FOR I := 1 TO ENDIDEN DO
001732  7                                  IF  IDENBUFF[I] <> BUFF[I]
001741  7                                  THEN  GOTO 3;
001751  7                             DISPOSE (PTRNEWREC);
001753  7                             GOTO 4;
001754  7                             END;
001754  8                      3: AUXPTR := PTR;
001756  6                      PTR := PTR^.NEXTSYM;
001763  6                      END;
001764  5               AUXPTR^.NEXTSYM := PTRNEWREC;
001772  5               END;
001772  4       WITH  PTRNEWREC^  DO
001777  4               BEGIN  (* IT IS NOT THERE, SO INSERT IT *)
001777  5               FOR I := 1 TO ENDIDEN DO
002002  5                    STRING[I] := IDENBUFF[I];
002022  5               LNGTH := ENDIDEN;
002024  5               NEXTSYM := NIL;
002025  5               END;
002025  4       4: END;
002026  3
002026  3  BLNKS:  (* RUN EITHER THE INDENTATION FUNCTION OR INCREMENT CODE LINES
002026  3          COUNTER *)
002026  3       BEGIN
002026  4       IF  NEWLINE
002026  4       THEN  IF  INDEX <> ENDLINE
002030  4             THEN  COUNT (INDNF);  (* COMPUTE INDENFUNC *)
002035  4       NEWLINE := FALSE;
002037  4       END;
002040  3
002040  3  ASSIGN:  (* DECREMENT   OPERATORS
002040  3                          OPERANDS
002040  3                          CODELINES *)
```

```
002040  3          BEGIN
002040  4          OPERATORS := OPERATORS - 1;
002042  4          OPERANDS  := OPERANDS - 2;
002044  4          CODELINES := CODELINES - 1;
002046  4          END;
002047  3
002047  3  EOLNM,
002047  3  EOLNSM,
002047  3  EOFM,
002047  3  EOFSM:     (* IGNORE PARAMETER *)
002047  3           BEGIN
002047  4           SCANNER;
002050  4           WHILE  ACTION = BLNKS  DO
002053  4                  BEGIN  (* SKIP BLANKS *)
002053  5                  NEWLINE := FALSE;
002054  5                  SCANNER;
002055  5                  END;
002056  4           IF  ACTION = LPAR
002056  4           THEN  WHILE  ACTION <> RPAR  DO
002063  4                        SCANNER  (* SKIP FILE NAME *)
002063  4           ELSE  YES := FALSE;
002070  4           END;
002071  3
002071  3  TERM:      (* TERMINATE CURRENT PROGRAM *)
002071  3          IF  ENDFLAG
002071  3          THEN  BEGIN
002073  4                MODFINI;
002074  4                MODINIT;
002075  4                END;
002076  3
002076  3  ENDLIST:    (* TERMINATE ACCUSE *)
002076  3           BEGIN
002076  4           KEY := PARACOUNT;  (* INSERT KEY VALUE *)
002100  4           MODULE := MODULE - 1;
002103  4           SORTPROGNOSIS (1,MODULE);  (* SORT ON THE KEY *)
002105  4           OUTPUTPROGNOSIS;           (* PRINT THE RESULTS *)
002106  4           (* UP TO SEVEN ADDITIONAL SORTS CAN BE REQUESTED *)
002106  4           MORESORTS (18,19,16,17,2,6,ADD(12,ADD(13,14)));
002131  4           PRINTFREQ;
002132  4           GOTO  1;
002134  4           END;
002135  3
002135  3  ABT:       (* ABORT THE CURRENT PROGRAM *)
002135  3           ABORT;
002137  3
002137  3  FUNCS,
002137  3  RECDEC,
002137  3  TOST,
002137  3  WITHST,
002137  3  RBRAC,
002137  3  EQUAL,
002137  3  COLON,
002137  3  COMMA,
002137  3  FINDOF1;
002140  3  DUMMY:      ENDFLAG := FALSE;
002142  3
002142  3  END;  (* CASE ACTION OF *)
```

```
002173  2  2794
002173  2  2795  END;  (* WHILE *)
002174  1  2796
002174  1  2797  END;  (* DRIVER *)
002222  0  2798
002222  0  2799
002222  0  2800
002222  0  2801  BEGIN (* MAIN PROGRAM *)
002222  1  2802  ACCUSEINIT;
000025  1  2803  SYMINIT;
000026  1  2804  SCANINIT;
000027  1  2805  DRIVER;
000030  1  2806  1:
000030  1  2807  END.  (* MAIN PROGRAM *)

          COMPILER-ESTIMATED "W" OPTION = 066135B.
```

# APPENDIX I

# ACCUSE USER MANUAL

ACCUSE EDITION 1, OCT 80

## 1. PURPOSE

ACCUSE PROCESSES A GROUP OF PROGRAMS AND MATCHES ANY PAIRS OF PROGRAMS THAT ARE SUFFICIENTLY SIMILAR SUCH THAT PLAGIARISM IS A POSSIBILITY. ACCUSE, HOWEVER, MAKES NO JUDGEMENT AS TO THE QUESTION OF PLAGIARISM. FINAL JUDGEMENT IS LEFT TO THE USER.

## 2. BACKGROUND

AS NOTED IN RECENT LITERATURE, PLAGIARISM HAS BECOME A PROBLEM IN INTRODUCTORY COMPUTER SCIENCE COURSES. ONE SOLUTION TO THIS PROBLEM HAS BEEN THE DEVELOPMENT OF A PROGRAM AT PURDUE UNIVERSITY TO QUANTIFY THE SAMENESS OF FORTRAN PROGRAMS. THIS PROGRAM UTILIZES SOFTWARE SCIENCE MEASURES OF LENGTH TO MEASURE SIMILARITY OF PROGRAMS. SOFTWARE SCIENCE WAS FOUNDED IN 1972 BY M. HALSTEAD. HE SUGGESTED FOUR BASIC PARAMETERS THAT ARE USEFUL MEASURES OF PROGRAM CHARACTERISTICS.

ACCUSE ATTEMPTS TO GO BEYOND THESE FOUR PARAMETERS IN THE BELIEF THAT ADDITIONAL PARAMETERS ARE AVAILABLE TO ESTABLISH DISSIMILARITY OF TWO OR MORE PROGRAMS. IT USES SEVEN PARAMETERS AND VARIOUS COUNTING HEURISTICS THAT RESULT IN THE COMPUTATION OF A CORRELATION NUMBER THAT IS USED TO DETERMINE THE SIMILARITY OF TWO PROGRAMS. ACCUSE MEASURES 20 PARAMETERS, WITH THE SEVEN THAT COMPRISE THE CORRELATION NUMBER SELECTED BY TESTING.

ACCUSE IS AN INFANT, AND SUGGESTIONS ARE INVITED FROM USERS AS TO ITS IMPROVEMENT.

## 3. GENERAL COMMENTS

ACCUSE WAS DESIGNED TO BE INEXPENSIVE: IT PROCESSES OVER 170 LINES PER SECOND. THE RESULT IS A COMPROMISE BETWEEN SPEED AND COMPREHENSIVE ANALYSIS.

CONSEQUENTLY, ACCUSE WILL NOT DISCOVER CHANGES MADE BY THE SOPHISTICATED PLAGIARIST. THIS IS RATIONALIZED WITH THE ASSUMPTION THAT THE STUDENT INTELLIGENT ENOUGH TO PLAGIARIZE WITH SOPHISTICATION HAS NO NEED TO PLAGIARIZE. HOPEFULLY, THOUGH, ACCUSE IS NOT SO SIMPLE-MINDED THAT IT IS EASY TO BEAT. IT IS MEANT TO MAKE PLAGIARISM WITHOUT DETECTION DIFFICULT TO ACHIEVE, AND IT IS MEANT TO DO THIS IN SUCH A WAY THAT ITS REPEATED USE DOES NOT COMPROMISE ITS HEURISTICS.

ACCUSE PRESENTLY MEASURES THE FOLLOWING 20 PARAMETERS:

1. TOTAL LINES
2. CODE LINES
3. CODE COMMENT LINES
4. MULTIPLE STATEMENT LINES
5. CONSTANTS AND TYPES
6. VARIABLES DECLARED (AND USED)
7. VARIABLES DECLARED (AND NOT USED)
8. PROCEDURES AND FUNCTIONS
9. VAR PARAMETERS
10. VALUE PARAMETERS
11. PROCEDURE VARIABLES (INCLUDES 9 AND 10)
12. FOR STATEMENTS
13. REPEAT STATEMENTS
14. WHILE STATEMENTS
15. GOTO STATEMENTS
16. UNIQUE OPERATORS
17. UNIQUE OPERANDS
18. TOTAL OPERATORS

19. TOTAL OPERANDS

20. INDENTING FUNCTION

THE SEVEN PARAMETERS USED TO CREATE THE CORRELATION NUMBER ARE:

1. UNIQUE OPERATORS
2. UNIQUE OPERANDS
3. TOTAL OPERATORS
4. TOTAL OPERANDS
5. CODE LINES
6. VARIABLES DECLARED (AND USED)
7. TOTAL CONTROL STATEMENTS

"TOTAL OPERATORS" DOES NOT INCLUDE ASSIGNMENT OPERATORS. ADDITIONALLY, FOR EVERY ASSIGNMENT OPERATOR FOUND, TWO OPERANDS ARE SUBTRACTED FROM "TOTAL OPERANDS," AND "CODE LINES" IS DECREMENTED. THIS WILL (HOPEFULLY) PREVENT ACCUSE FROM BEING MISLED BY INANE ASSIGNMENT STATEMENTS.

SINCE ACCUSE ONLY COUNTS VARIABLES, THE OBVIOUS TACTIC OF CHANGING VARIABLE NAMES MAKES NO DIFFERENCE TO ACCUSE. SINCE PASCAL REQUIRES DECLARATIONS, ACCUSE CAN KEEP TRACK OF VARIABLES DECLARED AND SUBSEQUENTLY USED OR NOT USED. HENCE EXCESS DECLARATIONS ARE AN INEFFECTIVE CHANGE TO A PROGRAM. "CODE LINES" IGNORES BLANK LINES, COMMENT LINES, AND DECLARATIONS. CONSTANTS OF ENUMERATED TYPES AND TAG FIELDS IN CASE CLAUSES OF RECORD DECLARATIONS THAT CONTAIN A DECLARATION ARE CONSIDERED VARIABLES. SINCE THESE CONSTANTS CANNOT BE READ OR WRITTEN, THEIR NON-USE IS NOTABLE.

4. HOW TO RUN ACCUSE

A. HOW PROGRAMS ARE IDENTIFIED

THE TOKTYP PROG IN PROCEDURE DRIVER GETS A UNIQUE IDENTIFIER FROM THE FIRST FOUR (CONSTANT CCIDLNGTH) CHARACTERS OF THE PROGRAM NAME. HENCE ANY PROGRAM

BEGINS

PROGRAM AJOBCOUNTCHANGE (INPUT,OUTPUT);

WHERE AJOB IS A UNIQUE IDENTIFIER ISSUED BY THE COMPUTING CENTER. THIS ENABLES EACH PROGRAM TO BE UNIQUELY AND EASILY IDENTIFIED.

B. INPUT OF PROGRAMS

INPUT IS A GROUP OF PROGRAMS INSERTED BACK TO BACK WITH NO SEPARATORS.

BECAUSE ACCUSE IS DESIGNED FOR 200 INPUT PROGRAMS, IT REQUIRES A 200 X 200 MATRIX (ALTHOUGH IT COULD USE A TRIANGULAR MATRIX IF ONE WERE AVAILABLE). THIS IMPLIES, OF COURSE, THE NEED FOR ALOT OF CORE. IF LESS PROGRAMS ARE TO BE INPUT, IT MAY BE ADVANTAGEOUS TO CHANGE THE CONSTANT MAXMODULE TO A SMALLER NUMBER AND THEN RECOMPILE ACCUSE.

C. RESULTS

ACCUSE PRINTS FOUR RESULTS FOR THE USER. THE FIRST IS A DUMP OF EACH INPUT PROGRAM'S CCID AND THE VALUES OF THE 20 PARAMETERS MEASURED BY ACCUSE. THIS DUMP IS SORTED ON THE INDENTING FUNCTION (SEE APPENDIX, "THE INDENTING FUNCTION").

THE SECOND RESULT IS A DUMP OF THE CCIDS AND THEIR RESPECTIVE VALUES OF THE SEVEN PARAMETERS USED TO COMPUTE THE CORRELATION NUMBER; EACH PARAMETER LIST IS SORTED SMALLEST TO LARGEST.

THE THIRD RESULT IS A FREQUENCY DISTRIBUTION GRAPH THAT INDICATES THE NUMBER OF PAIRS OF PROGRAMS WITH LIKE CORRELATION NUMBERS. THE MAX NUMBER OF * PRINTED FOR ANY CORRELATION NUMBER IS 40. THE TUKEY ESTIMATE FOR THE SUSPICION OF PLAGIARISM PRECEDES THE GRAPH.

THE FINAL RESULT IS A LIST OF ALL PAIRS OF PROGRAMS WITH CORRELATION NUMBER >= 28. TWENTY NINE IS CURRENTLY IDENTIFIED AS THE NUMBER THAT IMPLIES PLAGIARISM, WITH 3[illegible] THE MAXIMUM CORRELATION NUMBER POSSIBLE.

## 9. HOW ACCUSE WORKS

ACCUSE ASSUMES ALL PROGRAMS GIVEN TO IT ARE COMPILABLE. IF ACCUSE ABORTS FOR ANY REASON, IT FILLS THE PARAMETER LIST OF THE CURRENT PROGRAM WITH ZEROES. IT THEN FINDS THE FIRST "END" FOLLOWED BY A "." AND RESTARTS PROCESSING. AN ABORT, INCORRECT PROGRAM IDENTIFIERS, AND "WEIRD" COUNTS ARE USUALLY THE RESULT OF SHUFFLED INPUT CARDS.

ACCUSE LOADS ALL OF THE KEYWORDS AND DELIMITERS IN PASCAL (INCLUDING LOCAL IMPLEMENTATION KEYWORDS) INTO A SYMBOL TABLE. ANY ENTITY FOUND BY THE SCANNER IS FIRST TESTED BY PROCEDURE TEST TO SEE IF IT RESIDES IN THE SYMBOL TABLE. IF IT DOES, IT IS NOTED AS A KEYWORD. ELSE IT IS ASSUMED TO BE AN OPERAND.

WHEN A KEYWORD IS REDEFINED AS A VARIABLE BY THE USER, IT IS REMOVED FROM THE SYMBOL TABLE. IT IS REINSERTED INTO THE SYMBOL TABLE WHEN IT REGAINS ITS SYSTEM DEPENDENT DEFINITION.

ALL VARIABLES DECLARED IN A PROGRAM ARE PLACED ON A DECLIST AT THE APPROPRIATE NESTLEVEL AND REMOVED AS THEY ARE USED.

ALL NUMBERS USED IN THE PROGRAM ARE INSERTED ON A NUMLIST (IF NOT ALREADY THERE) AND ARE COUNTED AS UNIQUE OPERANDS.

LABEL AND VALUE DECLARATIONS ARE IGNORED.

THE NUMBER OF TYPES AND CONSTANTS IS COUNTED. THE REST OF THE DECLARATION IS IGNORED.

EXCEPT FOR THE VARIABLES THEMSELVES, VAR DECLARATIONS ARE IGNORED.

IF THE USER DESIRES TO MAKE CHANGES TO ACCUSE, HE SHOULD REFERENCE THE APPENDIX.

# APPENDIX J

## APPENDIX TO USER MANUAL

APPENDIX TO ACCUSE USER MANUAL

COMPONENTS AND HOW TO CHANGE THEM

A. INSERT/DELETE KEYWORDS FROM THE SYMBOL TABLE

CURRENTLY, 158 KEYWORDS AND DELIMITERS ARE CONTAINED IN THE SYMBOL TABLE. FOR EASE OF MODIFICATION, THE VALUE STATEMENT, AVAILABLE ON OUR COMPILER, IS USED TO INSERT KEYWORDS INTO THE SYMBOL TABLE. IN ADDITION, THE KEYWORDS ARE ARRANGED IN ALPHABETICAL ORDER. WHILE NOT EFFICIENT, AND NOT RECOMMENDED FOR A PRODUCTION MODEL, ALPHABETIZING MAKES FOR EASY CHECKING FOR THE PRESENCE OF A PARTICULAR ENTITY. TRANSLATION OF THE VALUE STATEMENT IS NOT DIFFICULT AND SHOULD NOT AFFECT PORTABILITY.

AN INSERT INTO (DELETE FROM) THE SYMBOL TABLE:

1. CHANGE THE CONSTANT NUMRESWORDS AS APPROPRIATE TO REFLECT THE CHANGE.

2. UNDER THE VALUE STATEMENT INSERT/DELETE THE KEYWORD AS APPROPRIATE, INITIALIZING ALL FIELDS TO THE DESIRED VALUES (SEE TYPE RESSYMBOL).

ACCUSE REMOVES REDEFINED KEYWORDS FROM THE SYMBOL TABLE UNTIL THEY REGAIN THEIR SYSTEM DEFINITION. WHEN REINSERTED INTO THE SYMBOL TABLE, THE KEYWORD IS PLACED AT THE END OF THE APPROPRIATE BUCKET IN THE SYMBOL TABLE. THIS IS DONE IN ANTICIPATION OF AN ORDERING OF KEYWORDS IN THE SYMBOL TABLE AND A DESIRE TO PRESERVE THAT ORDERING (IT IS ASSUMED RESERVED WORDS WILL BE FIRST IN THE BUCKET).

B. THE SCANNER

THE SCANNER IS A MODIFICATION OF THE PASCAL-J SCANNER AVAILABLE TO STUDENTS FOR THEIR WORK HERE AT THE UNIVERSITY OF COLORADO.

C. MAKING A NEW COUNT (OR CHANGING ONE)

ANY CHANGES TO A PARAMETER MUST BE RESOLVED IN PROCEDURE INSERTPROGNOSIS WHERE PARAMETERS ARE INSERTED INTO THE PROGNOSIS ARRAY AND IN PROCEDURE HEADINGS WHERE HEADINGS FOR THE VARIOUS PARAMETERS ARE OUTPUT. IF THE CHANGE INVOLVES ONE OF THE SEVEN PARAMETERS USED TO CALCULATE THE CORRELATION NUMBER, SEE SECTIONS E AND F.

MAKING A NEW COUNT WILL INVOLVE A CHANGE TO THE DRIVER (WHICH PROVIDES THE CODE TO CALCULATE COUNTS) AND TO PROCEDURE COUNT WHERE THE COUNTS ARE MADE. ANY CHANGE TO DRIVER IMPLIES WRITING CODE. IT IS SUGGESTED THAT THE USER CHECK THE PROGRAM TO MAKE SURE HE IS NOT REINVENTING THE WHEEL WHEN HE WRITES HIS CODE.

MAKING A NEW COUNT USUALLY INVOLVES THE CREATION OF A NEW TOKTYP TO BE RECOGNIZED BY DRIVER. THE TOKTYP MUST BE INSERTED INTO THE SCANNER OR INTO THE TOK FIELD (SEE TYPE RESSYMBOL) OF SOME KEYWORD SO THAT THE APPROPRIATE ACTION WILL BE PERFORMED WHEN RECOGNIZED. A NEW COUNTCLASS CAN BE CREATED UNDER THE TYPE STATEMENT AND THE NECESSARY CODE PLACED UNDER THE COUNTCLASS IN PROCEDURE COUNT.

IF THIS NEW COUNT INCREASES THE NUMBER OF PARAMETERS COUNTED, THE CONSTANT PARACOUNT MUST BE INCREMENTED (OR DECREMENTED IF A COUNT IS REMOVED). IN ADDITION, IF THE COUNT IS TO BE MADE FOR EACH INDIVIDUAL MODULE, THE COUNT MUST BE INITIALIZED IN MODINIT. IF THE COUNT WILL PREVAIL OVER THE ENTIRE EXECUTION OF ACCUSE, IT MUST BE INITIALIZED IN ACCUSEINIT.

D. THE INDENTING FUNCTION

THE INDENTING FUCTION IS CURRENTLY A SIMPLE ALGORITHM THAT COUNTS THE NUMBER OF LEFT, RIGHT, AND ZERO INDENTATIONS IN THE PROGRAM CODE. A SIGNIFICANT INDEN-

TATION (CONSTANT SIGINDENT) IS CONSIDERED TO BE AN INDENTATION OF AT LEAST THREE CHARACTERS. ONE OR TWO CHARACTER INDENTATION IS CONSIDERED AN ERROR THE STUDENT DID NOT DETECT OR FAILED TO CORRECT.

CURRENTLY, THE CONTENTS OF THE PROGNOSIS ARRAY IS DUMPED SORTED ON THE INDENTING FUNCTION. IT WAS INITIALLY THOUGHT THAT THE INDENTING FUNCTION WOULD PLAY AN IMPORTANT ROLE IN IDENTIFYING PLAGIARISM, BUT THIS DID NOT PROVE TO BE THE CASE. IF ALL PROGRAMS WERE PROCESSED THROUGH SOME SORT OF "PRETTY PRINTER," THE INDENTING FUNCTION COULD BECOME IMPORTANT. THIS ADDITIONAL COST IS CONSIDERED PROHIBITIVE AND IS CONTRARY TO THE INTENT OF ACCUSE BEING INEXPENSIVE TO USE.

E. CHANGING THE PARAMETERS TO COMPUTE THE CORRELATION NUMBER

ENDLIST IN DRIVER CALLS PROCEDURE MORESORTS. IT IS THIS CALL THAT DECIDES WHICH PARAMETERS WILL BE USED IN THE COMPUTATION OF THE CORRELATION NUMBER. PROCEDURE MORESORTS SORTS AND USES THE PARAMETERS SPECIFIED. WHILE MORESORTS REQUIRES SEVEN ARGUMENTS, THE DESIRE TO USE FEWER ARGUMENTS IS EFFECTED BY USING A ZERO FOR ONE OF THE ARGUMENTS. THE FUNCTION ADD IS PROVIDED TO SUM VARIOUS PARAMETERS. A WARNING IS PROVIDED WHEN ADD IS USED TO REMIND THE USER THAT THE HEADING PRINTED IS NOT NECESSARILY THE CORRECT ONE. CURRENTLY, "FOR STMT" IS THE HEADING PRINTED WHEN THE TOTAL NUMBER OF CONTROL STATEMENTS IS OUTPUT FOR THE USER.

IF YOU WANT TO CHANGE THE COMPUTATION OF THE CORRELATION NUMBER, SEE SECTION F.

F. CHANGING THE CORRELATION NUMBER

THE CURRENT CORRELATION NUMBER SCHEME IS ONLY TENTATIVE AND MAY VERY WELL BE SUBJECT TO IMPROVEMENT BY A MORE ELABORATE COMPUTATION SCHEME OR BY SIMPLE

CHANGES TO IMPORTANCE FACTORS (SEE PROCEDURE STATISTICS). THE CURRENT CORRELATION SCHEME IS CONTAINED IN STATISTICS AND PRINTED BY PRINTFREQ. ANY LARGE CHANGES TO THE CORRELATION SCHEME WILL REQUIRE REWRITING THESE TWO PROCEDURES.

THE CURRENT SCHEME INVOLVES COMPUTING AN INCREMENT FOR EACH PAIR OF PROGRAMS BASED ON THE EQUATION

INCREMENT = IMPORTANCE - (PCOUNTA - PCOUNTB)

WHERE PCOUNTA AND PCOUNTB REPRESENT PARAMETER COUNTS AND (PCOUNTA - PCOUNTB) <= WINDOW. IT SHOULD BE OBVIOUS THAT IMPORTANCE MUST BE > WINDOW. THIRTY TWO IS THE MAXIMUM CORRELATION NUMBER.

IF 29 IS THE NUMBER WHERE SUSPICION OF PLAGIARISM OCCURS, NO WINDOW SIZE NEED BE LARGER THAN THREE. HOWEVER, BECAUSE 29 IS ONLY AN OBSERVED PHENOMENON, AND ACCUSE IS STILL IN THE TEST PHASE, THE WINDOW SIZES HAVE BEEN LEFT AS ORIGINALLY IMPLEMENTED.

DATE
ILMED
-8