LA-5410-MS

INFORMAL REPORT

Dr - 397

175
10,20,75

ANSIFT:

An ANSI Standard FORTRAN Sifting Program

los alamos
scientific laboratory
of the University of California
LOS ALAMOS, NEW MEXICO 87544

UNITED STATES
ATOMIC ENERGY COMMISSION
CONTRACT W-7405-ENG. 36

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

In the interest of prompt distribution, this LAMS report was not edited by the Technical Information staff.

Printed in the United States of America. Available from
National Technical Information Service
U. S. Department of Commerce
5285 Port Royal Road
Springfield, Virginia 22151
Price: Printed Copy \$4.00; Microfiche \$0.95

LA-5410-MS
Informal Report
UC-32

ISSUED: October 1973

ANSIFT: An ANSI Standard FORTRAN Sifting Program

by

Clarice W. Cox

This work supported by the US Atomic Energy Commission's Division of
Reactor Research and Development (Reactor Physics Branch).

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

MASTER

Key

ANSIFT:
An ANSI Standard FORTRAN Sifting Program

By
Clarice W. Cox

ABSTRACT

ANSIFT looks for and changes or flags common nonstandard usages in CDC FORTRAN source decks. The program itself is written in standard FORTRAN to make it more adaptable for use on other computers.

I. INTRODUCTION

ANSIFT (from American National Standards Institute FORTRAN) examines source decks that are compilable on either the CDC 6600 or 7600 computers for some common nonstandard usages. It can detect, and sometimes correct, the usages listed below. These nonstandard usages hinder the adaptation of CDC FORTRAN codes to other computers.

1. Nonstandard characters (e.g., π),
2. Blank cards,
3. Nonstandard input/output statements (e.g., PRINT),
4. Use of unsubscripted array names or implied DO-loop notations in DATA initialization statements,
5. Unspecified repeat count or missing field separators in FORMAT statements,
6. ENCODE statements,
7. Two-branch logical IF statements,
8. Simple mixed-mode relational expression in logical IF statements,
9. Calls to system subroutines,
10. Seven-character variable names,
11. Hollerith replacement statements (e.g., variable = nH... or variable = nL...), and
12. Illegal subscript expressions (e.g., V(2*J+1)).

FORTRAN is a programming language commonly used in the computation of scientific research problems. In principle, any FORTRAN code should be readable by

any computer capable of compiling FORTRAN. But in fact, it usually cannot because local dialects which are incompatible with each other have been developed for each computer. Consequently, a code written in CDC FORTRAN is not usually compilable on an IBM-360 computer, and vice versa. Indeed, it is just those features which take advantage of a computer's unique capabilities that can make a code impossible to run on a different computer.

When a code has been produced that is of enough general interest for an outside installation to want to import it, the importer can be faced with a time consuming translation problem. This problem is greatly diminished if the code being exported has been written in ANSI Standard FORTRAN.¹ This language is a subset of FORTRAN IV that is common to the different major computers. With a few exceptions, this standard FORTRAN may be called the intersection of the various FORTRAN IV dialects.

ANSIFT was written to help the exporter of CDC-FORTRAN codes to standardize the language used and thus ease the translation problem. First, a working code was examined to find out what kinds of nonstandard usages it contained. Next, a control program was written that calls special subroutines to check for each kind of usage. Finally, the special subroutines were written and incorporated into the code. If a new nonstandard usage is to be added to the

list, a special subroutine to check for that usage can easily be added.

ANSIFT was itself written in ANSI Standard FORTRAN to make it exportable. It has checked itself as well as several large source programs. The test program in the appendix illustrates its capabilities.

II. USAGE

Four files are used - two besides the usual I/O.

<u>File Name</u>	<u>Description</u>
FSET1	Input file. Contains the source deck to be checked followed by a terminal card.
FSET2	Output file. Contains error messages followed by a listing of FSET4.
FSET3	Final version punch file. Contains card images ready for punching. Early in the program it is also used as a scratch file.
FSET4	Final version print file. Contains changed card images, if changes were made, as well as appropriate flags beyond column 80.

To use ANSIFT, the source deck to be checked must first be set up with a terminal card (a card containing the letter T in column 1) following its last card. This deck should be available to ANSIFT on the input file(FSET1). ANSIFT will not change the order of lines and statements in the source deck. However, it may delete or insert some lines, so the size of the output version of the source deck may differ from that of the input version.

There are two stages of processing. In the first stage, the input file (FSET1) containing the source deck is checked line by line. The accepted lines are written on a scratch file (FSET3). Program unit headlines are written on the output file (FSET2) to aid in locating errors. Error messages are then written on the output file following the headline of the routine in which they were detected.

In the second stage, the scratch file is read and statements are formed. Each statement is checked for nonstandard usages. Where possible, a nonstandard statement is changed to make it standard. Where it cannot be changed, the statement is flagged. The final version of the statement will be written onto a print file (FSET4). This file will contain, in addition to the corrected statements, all of the

comment lines, unchanged statements, and END lines of the original deck in their original order. The flags will also be written on this file beyond column 80 of the card image.

The listed output (FSET2) from this stage will consist, as in the first stage, of headlines of each program unit followed by error messages. At the end of the second stage, the final print version (FSET4), including flags, is copied onto the output file. Then the 80-column card images, without flags, are copied onto the final version punch file (FSET3).

III. PROGRAM DESCRIPTION

Two kinds of program units make up the ANSIFT code; special purpose routines and general purpose routines. The special purpose routines are concerned with a particular job or certain kind of statement. For example, FINCHK will only check statements beginning with the letters FORMAT. These routines are called by only one other routine.

The general purpose routines work with a statement image array or a section of the array and are not limited to a particular kind of statement. NNBLNK, for example, searches a portion of the array for a nonblank character. When it is found, KNECHR is sometimes called to tell what kind of nonblank character it is. These routines are freely called by other routines.

Within each of these two categories, there is a hierarchy of control. LINORD and STMTCK are major special purpose control routines called by ANSIFT, the main program. LGLSBS is a major general purpose control routine. These chains of command are illustrated in the flow chart (Fig. 1)

In the first stage of the processing, ANSIFT checks FORTRAN lines. There are four standard kinds of lines.

1. Initial line. Has only a statement label or blanks in columns 1-5, has blank or zero in column 6, and is not an END line. ANSIFT requires a nonblank character somewhere in columns 7-72.
2. Continuation line. Has a character other than letter C in column 1 and a character other than blank or zero in column 6. ANSIFT requires blanks in columns 1-5.
3. Comment line. Has letter C in column 1.
4. END line. Has blanks in columns 1-6 and the letters END appearing once each and in that order

Fig. 1. Simplified flow diagram for ANSIFT code.

FMTCHK Expansion

HOLRPL Expansion

in columns 7-72. All the rest of the columns are blank.

LINORD calls LINTYP to classify each line in the source deck. If the line is completely blank or is blank except for a zero in column 6, LINORD will delete the line and call ERRMSG to write the error message, "Blank line deleted". If the line is unrecognizable it will also be deleted with an error message written. All of the accepted lines are counted and written onto a scratch file, NF1 (FSET3). The terminal line (with letter T in column 1) signals the end of input. It will not be transferred to NF1. If the line preceding the terminal line is not an END line, LINORD will insert an END line and count it. LINTYP will substitute a + for = in column 6 of a continuation line.

In addition to checking the type of line, LINORD checks if lines are in the right order. If an END line is followed by any kind of line other than an initial or terminal line, the line will be deleted and an error message written. Excessive (more than 19 consecutive) continuation lines will be deleted. If a continuation line follows a comment line it will not be deleted but an error message will be written. In the second stage of processing, NF1 is rewound and read, one line at a time. Comment lines and END lines are transferred to the final version print file, NF2 (FSET4), without flags or changes. A single initial line, or an initial line followed by up to 19 continuation lines, is read into a statement image array for processing. SIMTCK examines this array to determine what kind of statement it is and then calls an appropriate routine to inspect it further.

<u>Starting Characters</u>	<u>Routine Called</u>
PRINT or READ	IOCHK
DATA	DATAACK
FORMAT	FMTCHK
ENCODE	ENCDCK
IF	IFCHK
CALL	SYSUB
None of the above	HOLRPL

In this stage a statement may be flagged or changed. Sometimes, the change requires the insertion of extra continuation lines or a new statement. There are four kinds of statements in which nonstandard usages will be changed, DATA, FORMAT, two-branch logical IF, and Hollerith replacement statements. In a DATA

statement, if the list of constants contains a Hollerith constant of the form nL... , the letter H will replace the letter L, resulting in nH... . In a FORMAT statement, a comma will be inserted if it was omitted following either a Hollerith or spacing (nX) specification, while a comma preceding or following a slash will be deleted. An asterisk-enclosed Hollerith specification will be replaced by an nH... specification. A parenthetical group without a repeat count will be changed in one of two ways. If the group is followed by a format specification, the number 1 is inserted as a repeat count. If the group terminates the FORMAT statement, the number 60000 is inserted as a repeat count and a slash is inserted before the group's final right parenthesis. A two-branch logical If such as IF (...) n₁,n₂ will be changed to the two statements

```
IF (...) GOTO n1
      GOTO n2      .
```

A Hollerith replacement statement of the form

variable = nH... or variable = nL...

will be replaced by the two statements

```
DATA dummy / nH... /
      variable = dummy      .
```

The dummy name is either IIIId (for integer type) or HHHHdd (for real type) where d is a decimal digit.

With the exception of the unlimited repeat count correction in a FORMAT statement, none of these changes will be explained in an error message. However, the final version of the statement will be flagged "CHANGED" or "INSERTED".

The flags, which are written on NF2 along with the first line of the final statement but beyond column 80, point out whether it is "NOT ANSI", "INCOMPLETE", "POSSIBLY NOT ANSI", "CHANGED", or "INSERTED". The headline of each routine is identified by the flag "NEW PROC". If the statement passes all the tests it is written on NF2 without flags.

All of the statement-checking routines write explanatory error messages on the output file if a nonstandard usage is detected. If an unspecified repeat count is detected in a FORMAT statement, the statement will be changed on NF2 and the changes will be described on the output file. All of the

error messages start by identifying the line or statement in which the error was detected. The rest of the message tells what was wrong. The possible explanations are listed in Table I, along with the names of the routines calling for them.

At the end of the second stage, the final version print file, NF2, is written on the output file. NF1 is rewound and the final version 80-column card images are written on it.

All of the routines in ANSIFT are listed and described in Table II.

TABLE I

ERROR EXPLANATIONS AND DETECTING ROUTINES			
Error Message	Routine	Error Message	Routine
Only an initial line is acceptable here. Illegal line deleted.	LINORD	Contains an implied DO-loop, an array name, or a Hollerith constant with an R Specification.	DATAACK
Unrecognizable line deleted.	LINORD	A non-ANSI DATA initialization statement.	DATAACK
Continuation line illegal here.	LINORD	No available correction space -- non-ANSI asterisk (*) specification.	STAR
Blank line deleted.	LINORD	No available correction space -- missing field separator.	FLDSEP
Missing END line inserted.	LINORD	No available correction space -- unlimited repeat count.	FMTCHK
Incomplete statement.	IOCHK, LISTV, LISTC, FMTCHK, SQUASH, ENCDCK, IFCHK, and IVARCK	A two-branch logical IF statement.	IFCHK
Seven-character variable name.	IVARCK and HOLRPL	Illegal subscript expression.	HOLRPL
Non-standard I/O statement.	IOCHK	A non-ANSI simple mixed mode expression.	IFCHK
Illegal character or illegal use of this character...	DATAACK, LISTC, and HOLRPL	Has a call to a system subroutine.	SYSUB
The number of real or integer variables found in Hollerith replacement statements exceeds 99.	HOLFIX	Illegal instruction.	INCDCK
Non-standard Hollerith replacement or assignment statement. (with an R specification).	HOLFIX	The preceding has either a non-standard O, or non-standard R specification.	FMTCHK
An unrecognizable IF statement form.	IFCHK	The preceding uses an unlimited repeat count...and is corrected as shown above.	FMTCHK
		May not be an ANSI IF.	IVARCK

TABLE II

DESCRIPTION OF PROGRAM UNITS IN ANSIFT CODE

Name	Description	Name	Description
	Routines in this section are special purpose control or checking routines.	LINORD	Called by ANSIFT to check the type and order of FORTRAN lines for deviations from ANSI standards. May insert or delete lines. Calls LINTYP and ERRMSG. Reads FSET 1 and writes FSETs 2 and 3.
ANSIFT	The main routine. Controls overall flow of sifting process. Calls LINORD to check lines and STMTCK to check statements. Writes FSETs 2, 3, and 4. Reads FSETs 3 and 4.		

TABLE II (cont.)

Name	Description	Name	Description
LINTYP	Checks a single line according to ANSI standards and assigns a type number to it. May change a continuation line. Called by LINORD and calls COMPAR.	IRELOP	Called by IFCHK to locate a relational operator. Calls NNBLNK.
STMTCK	Called by ANSIFT to check a complete statement for deviations from ANSI standards. Calls COMPAR to classify the statement and calls IOCHK, DATAK, FMTCHK, ENCDCK, IFCHK, SYSUB, or HOLRPL to check it. Also calls NNBLNK.	KTYPE	Called by IFCHK to locate a constant and identify its type. Calls NNBLNK and JDIGIT.
IOCHK	Called by STMTCK following detection of the word PRINT or READ to determine if the statement is a PRINT or nonstandard READ statement. Calls NNBLNK, JDIGIT, and ERRMSG.	SYSUB	Called by STMTCK following detection of the word CALL to determine if the statement is a call to a system subroutine. Calls COMPAR and ERRMSG.
DATAK	Called by STMTCK following detection of the word DATA to determine if the statement is a standard DATA initialization statement. May change a statement to make it standard. Calls LISTV, LISTC, ERRMSG, and NNBLNK.	HOLRPL	Called by STMTCK to check a statement not already classified as a nonstandard I/O, DATA FORMAT, ENCODE, IF or system subroutine CALL statement. It checks for a seven-character variable name, illegal subscripts, and whether the statement is a Hollerith replacement statement. May change a statement to make it standard. Calls LANSTR, LGLSBS, NNBLNK, KNBCHR, COMPAR, JDIGIT, ERRMSG, and HOLFIX.
LISTV	Called by DATAK to examine DATA statement list of variables for nonstandard usages such as implied DO-loop. Calls NNBLNK, JDIGIT, and ERRMSG.	HOLFIX	Called by HOLRPL to correct Hollerith replacement statements. Calls ERRMSG.
LISTC	Called by DATAK to examine DATA statement list of constants for nonstandard usages such as Hollerith constants specified by nR or nL. Calls NNBLNK, JDIGIT, and ERRMSG.	Routines in the following section are general purpose control or service routines.	
FMTCHK	Called by STMTCK following detection of the word FORMAT to determine if the statement is a standard FORMAT statement. May change a statement to make it standard. Calls LGLSBS, NNBLNK, SQUASH, JDIGIT, ERRMSG, STAR, FLDSEP, and URC.	SPACHK	Shifts the remaining portion of a given array to the right to make room for corrections. Called by STAR, FLDSEP, and URC.
SQUASH	Called by FMTCHK to eliminate unnecessary blanks from a FORMAT statement. Calls KNBCHR, JDIGIT, and ERRMSG.	DGTSTR	Encodes an integer and stores it in an array with elements in IH... format. Called by STAR and IFCHK.
STAR	Called by FMTCHK to change asterisk-enclosed Hollerith messages to the standard nH... specification. Calls SPACHK, DGTSTR, and ERRMSG.	LGLSBS	Checks whether a given array contains a standard subscript. Called by FMTCHK, IVARCK, and HOLRPL and calls LGLSSX.
FLDSEP	Called by FMTCHK to insert a comma following nH... or nX specifications. Calls SPACHK and ERRMSG.	LGLSSX	Called by LGLSBS to check whether a given array contains a standard subscript expression. Calls JDIGIT and LANSTR.
URC	Called by FMTCHK to correct unspecified repeat count. Calls NNBLNK and SPACHK.	LANSTR	Checks whether a given array contains an alphanumeric character string beginning with a letter. Called by IVARCK, HOLRPL, and LGLSSX and calls NNBLNK and KNBCHR.
ENCDCK	Called by STMTCK following detection of the word ENCODE to determine if the statement is an ENCODE statement. Calls NNBLNK and ERRMSG.	JDIGIT	Checks whether a given array contains a digit string. Called by IOCHK, LISTV, LISTC, FMTCHK, SQUASH, IFCHK, KTYPE, HOLRPL, and LGLSSX. Calls NNBLNK and KNBCHR.
IFCHK	Called by STMTCK following detection of the word IF to determine if the statement is a two-branch logical IF or if it contains a simple mixed-mode relational expression. May change a statement to make it standard. Calls NNBLNK, JDIGIT, DGTSTR, ERRMSG, IVARCK, IRELOP, and KTYPE.	KNBCHR	Tells which of five kinds of characters a given nonblank character is. Called by SQUASH, HOLRPL, LANSTR, and JDIGIT.
IVARCK	Called by IFCHK to locate a variable name. Calls LANSTR, LGLSBS, NNBLNK, and ERRMSG.	COMPAR	Checks whether a given array contains a FORTRAN keyword. Called by LINTYP, STMTCK, SYSUB, and HOLRPL and calls NNBLNK.
		NNBLNK	Checks whether a given array contains a nonblank character. Called by several routines.
		ERRMSG	Writes an error message on the output file. Called by several routines.
		BLOCK DATA	Sets initial values of variables in labeled COMMON blocks.

IV. REFERENCES

1. American National Standard FORTRAN, USAS X3.9-1966, American National Standards Institute (1966).

APPENDIX

ANSIFT TEST PROGRAM

I N P U T

```

PROGRAM TEST(INP,OUT)
IMPLICIT INTEGER (A-X)
COMMON B,IF
DIMENSION IF(5)
DIMENSION A(10),GOTO (1,1,1),FORMAT(1,2,3),GOTOIT(2,2),B(10),
C*CRE TO COME
      C(10),C(3)
      IF(B.EQ.0) GOTO 50
      ENCCCE(20,20,A)A
28  FORMAT(10R1,2H 4X*HOLLERITH FIELD*)
29  FORMAT(* THIS AN INCORRECT ONE*020)
161  FORMAT(25X25X3H . (20A6))
50  FORMAT(3,2,1)=6
      GOTO (20,30,40,50), I
20  GOTO (20,30,40,50) I
30  GOTO (1,2,3)=I
40  IF(2-EQ-2) = K

      IF(2.NE.2) 20,24
      IF(Z+2..NE.0) 255,20
23  IF(TWC.EQ.2) GOTO (30,40) J
      CALL SECCNO(GLORP)
      CALL TIMER(ARGJMNT)
      CALL CLCK
      CALL BUFOOPEN(MORE)
      CALL CLOCK(LESS)
24  GOTO (1+J)=1+J
      GOTO (2+I)=1+J
      IF(PRINT(5,ARG,1+3,4).NE.0) 40,50
3   GOTOIT(I+2,K*3)=PRINT(5,ARG,I*3+J,1)
7   FORMAT(15H*****TEST IT)
      DATA B(2),B(3),B(4)/1,2,3/
C
C  THESE ARE THE FIRST NODES...
C
      ,B(5),B(6),C /8,9,4,6/
+
C  THATS ALL
      DO 2 W+CM=1
      *
      DO 4 W+CM=1,10,2
      DO 1 MCRE=6,K,M
1   CONTINUE
C   THIS IS A COMMENT
      +
4   CONTINUE
2   CALL SEGMENT(GLORP2)

```

```

      KL2=9L3+J=I*KL
999  KL2=10+ * $ = /
      CQ1=2R .
      DATA C/25H
      DATA (C(I), I=4,6)/ 3*4H****/
      IF (INTGRT F.GT.2.576) STOP
      IF (INTGRT -0.23.GE.0.6) RETURN
      IF (INTGRL(A))=5
      IF (I)=INTGRLA
      GOTO 255
255

      $ CONTINUE
C
      END

C  HERE IS THE REST OF THE LAST COMMENT
C
      SUPRCUTINE GOTO (A,B,C,C)
      IF (SUM.LT.PRITE) 255,250
      TEST2=3HTO1
      NTEST=4HTO1B
250  RETURN
255  STCF
T

```

O U T P U T

```

      PROGRAM TEST(INP,CUT)

*****ERROR FOUND IN--
*
*      ' C(10),C(3)
*
*****CONTINUATION LINE ILLEGAL HERE.

*****ERROR FOUND IN--
*
*
*
*****BLANK LINE DELETED.

*****ERROR FOUND IN--
*
*      ' ,B(5),E(6),C /8,9,4,6/
*
*****CONTINUATION LINE ILLEGAL HERE.

*****ERROR FOUND IN--
*
*
*
*****UNRECOGNIZABLE LINE DETECTED.

*****ERROR FOUND IN--
*

```

*
*
*****CONTINUATION LINE ILLEGAL HERE.

*****ERROR FOUND IN--
*
*
*
*****BLANK LINE DELETED.

*****ERROR FOUND IN--
*
*C HERE IS THE REST OF THE LAST COMMENT
*
*****ONLY AN INITIAL LINE IS ACCEPTABLE HERE. ILLEGAL LINE DETECTED.

*****ERROR FOUND IN--
*
*C
*
*****ONLY AN INITIAL LINE IS ACCEPTABLE HERE. ILLEGAL LINE DETECTED.

SUBROUTINE CCTC (A,B,C,D)

*****ERROR FOUND IN--
*
*T
*
*****MISSING END LINE INSERTED.

PASS ONE SUMMARY
9 ERROR(S) DETECTED
2 ILLEGAL LINE(S) DELETED
1 UNRECOGNIZABLE LINE(S) DELETED
3 ILLEGAL CONTINUATION LINE(S)
2 BLANK LINE(S) DELETED
1 A MISSING END LINE INSERTED

STATEMENT LISTING

PROGRAM TEST(INP,OUT)

*****ERROR FOUND IN--

*
*IF (B.EQ.0) GOTO 50
*

*****A NON-ANSI SIMPLE MIXED MODE EXPRESSION.

*****ERROR FOUND IN--

*
*ENCODE (20,20,A) A
*

*****ILLEGAL INSTRUCTION.

*****ERROR FOUND IN LINE 1 , POSITION 10 , IN THE FOLLOWING STATEMENT--

*
*FORMAT (10R1,2H 4X*HOLLERITH FIELD*)
*

*****THE PRECEDING HAS EITHER A NON-STANDARD D, OR NON-STANDARD R SPECIFICATION.

*****ERROR FOUND IN LINE 1 , POSITION 34 , IN THE FOLLOWING STATEMENT--

*
*FORMAT (22H THIS AN INCORRECT ONE,020)
*

*****THE PRECEDING HAS EITHER A NON-STANDARD D, OR NON-STANDARD R SPECIFICATION.

*****ERROR FOUND IN LINE 1 , POSITION 22 , IN THE FOLLOWING STATEMENT--

*
*FORMAT (25X,25X,3F . ,(20A6))
*

* THE PRECEDING USES AN UNLIMITED REPEAT COUNT--

*
*FORMAT (25X,25X,3F . ,60000(20A6/))
*

*****AND IS CORRECTED AS SHOWN ABOVE.

*****ERROR FOUND IN--

*
*GOTO (20,30,40,50) I
*

*****ILLEGAL SUBSCRIPT EXPRESSION.

*****ERROR FOUND IN--

*
*IF (2-E Q-2) = K
*

*****ILLEGAL SUBSCRIPT EXPRESSION.

*****ERROR FOUND IN--

*
*IF (T.D.EC.2) GOTO (30,40) J
*

*****A NON-ANSI SIMPLE MIXED MODE EXPRESSION.

*****ERROR FOUND IN--

*
*CALL SECCNC(GLORP)
*

*****HAS A CALL TO A SYSTEM SUBROUTINE.

```

*****ERROR FOUND IN--
*
*CALL BLFOPEN(MORE)
*
*****HAS A CALL TO A SYSTEM SUBROUTINE.

*****ERROR FOUND IN--
*
*CALL CLOCK(LESS)
*
*****HAS A CALL TO A SYSTEM SUBROUTINE.

*****ERROR FOUND IN--
*
*GOTO (I+J)=I+J
*
*****ILLEGAL SUBSCRIPT EXPRESSION.

*****ERROR FOUND IN--
*
*GOTO (2+I)=I+J
*
*****ILLEGAL SUBSCRIPT EXPRESSION.

*****ERROR FOUND IN--
*
*GOTOIT (I+2,K*3)=FRINT(5,ARG,I*3+J,1)
*
*****ILLEGAL SUBSCRIPT EXPRESSION.

*****ERROR FOUND IN--
*
*FORMAT (15H*****TEST IT)
*
*****INCOMPLETE STATEMENT.

*****ERROR FOUND IN--
*
*DO 2 WHON=1
*
*****SEVEN CHARACTER VARIABLE NAME.

*****ERROR FOUND IN--
*
*CALL SEGMENT(GLORP2)
*
*****HAS A CALL TO A SYSTEM SUBROUTINE.

*****ERROR FOUND IN LINE 1 , POSITION 6 , IN THE FOLLOWING STATEMENT--
*
*CQ1=2R .
*
*****NON-STANDARD HOLLERITH REPLACEMENT OR ASSIGNMENT STATEMENT--WITH AN R SPECIFICATION.

*****ERROR FOUND IN--
*
*DATA D/25H
*
*****CONTAINS AN IMPLIED DO-LOOP, AN ARRAY NAME, OR A HOLLERITH CONSTANT WITH AN R
SPECIFICATION.

```

```

*****ERROR FOUND IN--
*
*DATA  (C(1),I=4,6)/  3*4M****/
*
*****CONTAINS AN IMPLIED DO-LOOP, AN ARRAY NAME, OR A HOLLERITH CONSTANT WITH AN R
SPECIFICATION.

*****ERROR FOUND IN--
*
*IF (I*INTGRF.GT.2.576) STOP
*
*****SEVEN CHARACTER VARIABLE NAME.

*****ERROR FOUND IN--
*
*IF (I*INTGRF.GT.2.576) STOP
*
*****A NON-ANSI SIMPLE MIXED MODE EXPRESSION.

*****ERROR FOUND IN--
*
*IF (I*INTGRF-0.23.GE.0.6) RETURN
*
*****MAY NOT BE AN ANSI IF

*****ERROR FOUND IN--
*
*IF INTGRAL(A)=5
*
*****ILLEGAL SUBSCRIPT EXPRESSION.

SUBROUTINE COTC (A,B,C,D)

```

MLFILE LISTING

PROGRAM TEST(INP,CUT)	*****NEW PROC
IMPLICIT INTEGER (A-X)	NOT ANS I
COMMON B,IF	
DIMENSION IF(5)	
DIMENSION A(10),GOTO (1,1,1), FORMAT(1,2,3),GOTOIT(2,2),B(10),	
C MORE TO CCME	
* D(10),C(3)	NOT ANS I
IF (B.EQ.0) COTO 50	NOT ANS I
ENCODE(20,20,A)A	NOT ANS I
28 FORMAT(10R1,2F ,4X,15HOLLERITH FIELD)	NOT ANS I CHANGED
29 FORMAT(22F THIS AN INCORRECT ONE,020)	NOT ANS I CHANGED
161 FORMAT(25X,25X,3F , 60000(20A6/))	CHANGED
50 FORMAT(3,2,1)=6	
GOTO (20,30,40,50),I	
20 GOTO (20,30,40,50) I	NOT ANS I
30 GOTO (1,2,3)=1	
40 IF (2-EQ-2) = K	NOT ANS I
IF (2.NE.2) COTC 20	CHANGED INSERTED
GOTO 24	
IF (2+2.NE.0) COTC 255	CHANGED INSERTED
GOTO 20	
23 IF (2*0.EQ.2) COTC (30,40) J	NOT ANS I

	CALL SECOND(GLGRP)		ACT	ANS I
	CALL TIMER(ARGUMNT)			
	CALL CLKR			
	CALL BUFOFEN(MORE)		NOT	ANS I
	CALL CLOCK(LESS)		ACT	ANS I
24	GOTO (I+J)=I+J		ACT	ANS I
	GOTO (2+I)=I+J		ACT	ANS I
	IF(FRINT(5,ARG,I+3,4).NE.O) GOTO 40		CHANGED	INSERTED
	GOTO 50			
3	GOTOIT((I+2,K*3)=PRINT(5,ARG,I*3+J,1))		ACT	ANS I
7	FORMAT(15#11111TEST IT)		I	ACCMPLTE
	DATA E(2),E(3),E(4)/1,2,3/			
C				
C	THESE ARE THE FIRST NODES...			
C	* .P(5),B(6),C /8,9,4,6/		ACT	ANS I
	+		ACT	ANS I
C	THATS ALL			
	DO 2 WHQM=1		ACT	ANS I
	DO 4 WHQM=I,10,2			
	DO 1 PCFE=6,K,M			
1	CONTINUE			
C	THIS IS A COMMENT			
	+		ACT	ANS I
4	CONTINUE			
2	CALL SEGMENT(GLOPP2)		ACT	ANS I
	DATA IIIIO1/ 9H3+J=I*KL /		CHANGED	INSERTED
	K12=IIIIO1			
	DATA IIIIO2/10H + * \$ = //		CHANGED	INSERTED
999	K12=IIIIO2			
	CQ1=2R.		ACT	ANS I
	DATA D/25H	/	ACT	ANS I
	DATA (C(I),I=4,6)/ 3*4*****/		ACT	ANS I
	IF(INTGRAT.F.T.2.576) STOP		ACT	ANS I
	IF(INTGRAT-O.23.GE.O.6) RETURN	POSSIBLY	ACT	ANS I
	IF(INTGRL(A))=5		ACT	ANS I
	IF(1)=INTGRLA			
	GOTO 255			
255				
	* CONTINUE			
C	END			
	SUBROUTINE GOTC (A,B,C,D)		***** PROC	
	IF(SUM.(T.PRITE) GOTO 255		CHANGED	INSERTED
	GOTO 250			
	DATA HHHHO1/ 3HTO1 /		CHANGED	INSERTED
	TEST2=HHHO1			
	DATA IIIIO1/ 4HTO1B /		CHANGED	INSERTED
	NTEST=IIIIO1			
250	RETURN			
255	STOP			
	END			

PASS TWO SUMMARY

25	UNCORRECTED STATEMENT(S)
1	INCOMPLETE STATEMENT(S)
1	QUESTIONABLE USAGE(S)
11	CORRECTED LINE(S)
8	INSERTED STATEMENT(S)

PASS ONE SUMMARY

9 ERROR(S) DETECTED
2 ILLEGAL LINE(S) DELETED
1 UNRECOGNIZABLE LINE(S) DELETED
3 ILLEGAL CONTINUATION LINE(S)
2 BLANK LINE(S) DELETED
1 A MISSING END LINE INSERTED

TASK SUMMARY

70 CARDS PROCESSED
74 CARDS WRITTEN ONTO THE PUNCH FILE
2 ROUTINE(S) PROCESSED

-+- *-+-* *-+-* *-+-* *-+-* *-+-* *-+-*