There now exists a version of LISP 1.5 on the Dartmouth time-sharing system. It resides under system name "EVAL." EVAL is a modification of LISP 1.5 developed for the IBM 7090 at M.I.T. It is similar enough to the M.I.T. version that an adequate reference manual is:

LISP 1.5 PROGRAMMERS MANUAL
BY J. McCARTHY ET AL.
M.I.T. PRESS

However the user of EVAL must keep in mind the following differences between the system at Dartmouth and the system described in the manual.

(1) The LISP interpreter at Dartmouth is EVAL instead of the more usual EVALQUOTE. This means the user must make his program consist of forms for EVAL instead of doublets for EVALQUOTE.

For example:
CAR((A B C)) would become (CAR(QUOTE(A B C)))
CONS(A B) would become (CONS(QUOTE A)(QUOTE B))

However the execution of (INTERPRETQ T) will cause the interpreter to become EVALQUOTE. INTERPRETQ(() will bring back EVAL.

(2) The restriction against having an atom with a print name of more than 30 characters has been removed. The only present restriction is how much storage the user is willing to give for the saving of the print name.

(3) The symbols for true and false are T and NIL respectively. There is no *T* or *F* in the Dartmouth system. Both T and NIL are constants that evaluate into themselves.

(4) The form of numbers on input and output has been changed somewhat. On input a string of digits preceded by a minus sign or an optional plus sign that contains no decimal point (.) is read as an integer in the current input base. (For example 123 -456). A similar string of digits but followed immediately by only a decimal point is taken to be an integer in base ten. (For example 356. -794.). Floating point numbers take the
SAME FORM AS DESCRIBED IN THE MANUAL. IT IS STILL INCORRECT TO HAVE A NUMBER WITH A DECIMAL POINT AS THE FIRST CHARACTER AS THE DECIMAL POINT WILL BE INTERPRETED AS PART OF LISP DOT NOTATION. IF THERE IS AN "E" IN A FLOATING POINT NUMBER THE DECIMAL POINT MAY BE DROPPED. IF IT IS IT IS ASSUMED TO BE BEFORE THE "E". TO SET THE INPUT AND OUTPUT BASE OF NUMBERS TWO NEW FUNCTIONS WERE ADDED--SETIBASE AND SETJBASE. SETIBASE ONLY EFFECTS THE INPUT OF NUMBERS THAT DO NOT CONTAIN A DECIMAL POINT OR AN "E". SETJBASE DOES NOT EFFECT THE OUTPUT OF FLOATING POINTING POINT NUMBERS.

CONTINUED IN LISP-2***
(5) \textit{cond} has been redefined to accept different forms of arguments than the previous definition. The expanded definition does still however accept expressions coded in the old manner except that it returns nil instead of giving an error if none of the propositions are true. For a description of the new additions to \textit{cond} see \textit{cond-***}.

(6) The restriction that \textit{go} and \textit{return} may only appear on the top level of a \textit{prog} has been removed. Instead they may be used anytime after a \textit{prog} has been entered. When they are evaluated \textit{go} has as its value the point to be transferred to and \textit{return} has as its value the return value of the \textit{prog}. The execution of \textit{go} and \textit{return} is inhibited until the interpreter returns to the top level of the current \textit{prog} at which time \textit{prog} then transfers to the point specified by \textit{go} or \textit{return}. \textit{Go} and \textit{return} may only refer to the last \textit{prog} entered by the interpreter; they cannot refer to \textit{progs} entered previously to the current one.

(7) The section of the manual describing \textit{overlord} should be ignored. Instead one sets up his program as a collection of forms. Each of these forms will be evaluated and its result printed. Reaching the end of a \textit{prog} will cause an end of file error (error 4). The read routine completely ignores carriage returns and line numbers so that a form may run for several lines. There exists a special function called \textit{listen} that causes \textit{eval to interpret from the teletype instead of the current program.}

(8) LISP's error message's are described in \textit{lsperr***}.

(9) List structure is stored in the 235 differently than on the 7090. Each word of list structure takes two words of core memory in the 235. The even location word contains the car half and the next higher odd location word contains the cdr half. All list pointers point to the even half of the word. For this reason the atom head (the car of an atom) in the 235 is an odd pointer instead of the -1 that is used on the 7090. Because there is no tag portion in the 235 numbers are represented as atoms with the property fixnum or flonum instead of pname as with regular atoms. The property after the indicator fixnum is a pointer to an integer.
AND THE PROPERTY AFTER THE INDICATOR FLONUM IS A
POINTER TO A FLOATING POINT NUMBER. TO ACCOMODATE
PNAMEs AND NUMBERS FULL-WORD SPACE ALSO CONSISTS OF
DOUBLE WORDS--AN EVEN LOCATION AND THE NEXT HIGHER ODD
LOCATION.

(10) THERE EXISTS IN THE LIBRARY FOR THOSE WHO ARE
INTERESTED A COPY OF THE WANG ALGORITHM FOR
PROPOSITIONAL CALCULUS UNDER WANG--***.

(11) WHEN THE PROGRAMMER MAKES A CALL TO READ, THE
FUNCTION DOES NOT READ. THE FIRST LIST AFTER STOP BUT
INSTEAD READS THE NEXT LIST THAT HAS NOT BEEN READ BY
THE INTERPRETER.

(12) A LIST AND DESCRIPTION OF THOSE FUNCTIONs IN THE
DARTMOUTH LISP SYSTEM CAN BE FOUND IN LSPFNC***.

(13) THE INTERPRETER IS ACTUALLY AS DESCRIBED IN
APPENDIX B OF THE MANUAL EXCEPT THAT EVA
LQUOTE MUST BE
ASKED FOR SPECIALY AND EVAL IS THE NORMAL INTERPRETER.
ALSO THERE IS NO SPECIAL CHECK FOR NIL IN EVAL OR
APPLY. THIS MEANS THAT (APPLY (ARGS ALIST) CAUSES AN
ERROR MESSAGE SAYING NIL IS AN UNDEFINED FUNCTION.

(14) THERE ARE NO PROVISIONS FOR LAB OR A LISP COMPIlER
IN DARTMOUTH LISP.

(15) THE LISP READ ROUTINE DOES NOT THE RECOGNIZE THE
$S$ CONVENTION OF THE 7090 FOR QUOTING CHARACTERS IN
PRINT NAMES. INSTEAD ONE USES THE SLASH (/). THE NEXT
CHARACTER FOLLOWING A SLASH IS PLACED IN THE PNAME
REGARDLESS OF WHAT IT IS. CHARACTERS THAT MUST BE
SLASHED ARE <SPACE>, <COMMA>, <DOT>, <SLASH>, <RIGHT
PARENTHESIS>, <LEFT PARENTHESIS>, <CARRIAGE RETURN>, AND
LEADING <PLUS SIGN>, <MINUS SIGN>, AND <DIGITS> TO
PREVENT CONFUSION BETWEEN ATOMS AND NUMBERS.

(16) SINCE LISP IS A RAPIDLY CHANGING SYSTEM EACH
CHANGE WILL CARRY WITH IT A NUMBER WHICH INDICATES
WHICH VERSION IN CURRENTLY IN USE. THIS NUMBER IS
PRINTED AT THE BEGINNING OF THE RUN IN THE FORM OF
"LISP XX" WHERE XX IS THE UPDATE NUMBER. IN ADDITION
IF THERE IS AN E FOLLOWING THE NUMBER THIS INDICATES
THE CURRENT VERSION IS EXPERIMENTAL AND MAY CONTAIN
PROGRAMMING ERRORS. WITH EACH CHANGE IN VERSION A
NOTATION WILL BE MADE IN THE BEGINNING OF LSPUPD***
INDICATING THE ADDITIONS OVER PREVIOUS VERSIONS. IN
ORDER TO HELP IN MAKING THESE CHANGES AND ADDITIONS THE
USER IS ENCOURAGED TO APPEND QUESTIONS AND COMMENTS TO
FILE "LISP??" UNDER USER NUMBER 407886.
THIS IS A LIST OF THE ATOMS THAT ARE INITIALLY DEFINED IN THE DARTMOUTH VERSION OF LISP. A $ <DOLLAR SIGN> INDICATES A CHANGE FROM THE "LISP 1.5 PROGRAMMERS MANUAL."

LAMBDA

QUOTE [FSUBR]

CONS [SUBR]

EQ [SUBR].

LIST [FSUBR]

COND [FSUBR] $

COND HAS BEEN MODIFIED TO ACCEPT AN EXPANDED FORM OF PROPOSITIONS. SEE COND---*** FOR DETAILS.

CDR; CAR; CDDR; CDAR; CADR; CAAR; CDDDR; CDDAR; CDADR; CDAAR; CADDR; CADAR; CAADR; CAAR; [SUBR]

RPLACA; RPLACD; [SUBR] $

A CHECK IS MADE TO SEE IF THE USER IS MODIFYING A LOCATION THAT IS NOT IN FREE STORAGE OR FULL WORD SPACE. IF SO, THE OPERATION IS NOT DONE AND AN ERROR IS GIVEN. SINCE INITIAL DEFINITIONS ARE NOT IN THESE AREAS THE USER MAY NOT MODIFY THEM.

READ [SUBR] $

READ NOW READS THE NEXT UNREAD LIST FROM THE DISK. IT IS INITIALLY SET TO READ FROM YOUR CURRENT PROGRAM AREA.

PRINT [SUBR]

PRIN1 [SUBR] $

PRIN1 MAY NOW BE USED TO PRINT LISTS AS WELL AS ATOMS. IF GIVEN A LIST IT PRINTS IT NORMALLY BUT DOES NOT TERMINATE THE PRINT LINE.

TERPRI [SUBR]

PRG [FSUBR]

RETURN [SUBR] $

RETURN MAY NOW BE GIVEN ANYTIME AFTER A PRG HAS BEEN ENTERED. IF IT IS NOT GIVEN ON THE TOP LEVEL OF A PRG ITS ACTION IS INHIBITED UNTIL CONTROL RETURNS TO THE PRG.
G$ [FSUBR] $

G$ may be used anytime after a prog has been entered. If it is not given on the TJP level of a prog its action is inhibited until control returns to the prog.

G*$ [SUBR] $

G*$ is identical to G$ except that it evaluates its argument. It may be used for a switch-type transfer.

ADVANCE [SUBR] $

ADVANCE reads the next character from the disk. It ignores line numbers. ADVANCE should be used carefully as characters it reads will not be read by the READ function.

ADVANCE* [SUBR] $

ADVANCE* is identical to ADVANCE except that it does not ignore line numbers.

READTTY [SUBR] $

READTTY is a function of NJ arguments. Its value is one list typed in from the users teletype. It causes a call for input and continues to give one until a complete list is typed. This means that if parentheses do not balance on one line the computer will continue to ask for input until they do. As with READ, READTTY also ignores <carriage return>s. This means that a single atom must have a space typed after it. On any call for input there may be given more than one line; if so the extra lists (or part of a list) will be used on the next use of READTTY.

ADVANCETTY [SUBR] $

ADVANCETTY reads the next character from the users teletype. If the last character read was a <carriage return> then a call of ADVANCETTY causes a call for input and it returns the first character typed. The very first character read by ADVANCETTY will be a <carriage return>.

FIXNUM $)

The property of a fixed point number.

FLONUM $

The property of a floating point number.

PACK [SUBR]

CLEARBUFF [SUBR]

MAKNUM [SUBR] $

MAKNUM is a function of two arguments. It creates an atom with a property list of MAKNUM's second argument followed by MAKNUM's first argument. MAKNUM is often used as an easy way to create numbers from full words.

NUMJB [SUBR]
PLUS [FSUBR]
TIMES [FSUBR]
CONTINUED IN LSPFND***
A ' v·

[Image 0x0 to 611x792] [12x749], [40x749]' [68x751]/' [13x737],

A ' v·

[Image 0x0 to 611x792] [12x749], [40x749]' [68x751]/' [13x737],

A ' v·

[Image 0x0 to 611x792] [12x749], [40x749]' [68x751]/' [13x737],

A ' v·
DOES NOT WORK, TRY BACK OFF AND TRY ANOTHER PATH.

ERROR: [SUBR]

EQUAL [SUBR]

OR; AND [FSUBR]

NOT [SUBR]

PUTPROP [SUBR] $

PUTPROP IS DARTMOUTH'S VERSION OF LISP'S WAY OF
DEFINING THINGS. IT IS A FUNCTION OF THREE ARGUMENTS. THE
FIRST OF THESE MUST BE AN ATOM AND IT IS THIS ATOM THAT IS
BEING DEFINED. THE SECOND ARGUMENT IS THE DEFINITION AND
THE THIRD ARGUMENT IS THE DEFINITION TYPE (USUALLYEXPR OF
EXPR). ANY OTHER OCCURRENCE OF THE PROPERTY TYPE ON THE
DEFINED ATOM IS REMOVED.

DEFPROP [FSUBR] $

DEFPROP IS IDENTICAL TO PUTPROP EXCEPT THAT DEFPROP
QUOTES ITS THREE ARGUMENTS. THIS ALLOWS THE USER TO GIVE
ATOMS PROPERTIES WITHOUT TYPING QUOTE ALL THE TIME.

CSET [SUBR]

CSETQ [FSUBR]

PAIR [SUBR]

FUNARG

LABEL

T [APVAL] $

T REPLACES *T* IN THE LISP MANUAL. T IS A CONSTANT
THAT EVALUATES INTO ITSELF. PREDICATES IN LISP RETURN T OR
NIL.

NIL [APVAL]

GET [SUBR]

SASSOC [SUBR]

NCOUNT [SUBR]

EVAL [SUBR]

APPLY [SUBR]

APPLY NO LONGER MAKES A SPECIAL CHECK FOR NIL AS A
FUNCTION NAME. INSTEAD (APPLY NIL ARGS ALIST) WILL CAUSE AN
ERROR STATING THAT NIL IS AN UNDEFINED FUNCTION.
LISTEN  [SUBR]  $
LISTEN  IS  A  FUNCTION  OF  USE  FOR  TIME-SHARING  LISP.  ON
ENTERING  A  LISTEN  THE  INTERPRETER  STOPS  EVALUATING  FROM  THE
DISK  AND  INSTEAD  EVALUATES  STATEMENTS  FROM  THE  TELETYPE.
LISTEN  IS  TERMINATED  BY  TYPING  THE  ATOM  "STOP".  CARE  MUST
BE  TAKEN  TO  INSURE  THAT  STOP  IS  FOLLOWED  BY  A  SPACE  TO
PREVENT  THE  TIME-SHARING  SYSTEM  FROM  THINKING  THE  INPUT  IS  A
COMMAND  TO  STOP  LISP.  WHEN  LISTEN  IS  TERMINATED  IT  RETURNS
NIL.  THE  TYPICAL  LISP  PROGRAM  CONSISTS  OF  A  SET  OF  FUNCTION
DEFINITIONS  FOLLOWED  BY  A  LISTEN.

STOP  [SUBR],  $
STOP  AS  AN  ATOM  IS  USED  TO  TERMINATE  A  LISTEN.  STOP  AS
A  FUNCTION  IS  USED  TO  GIVE  A  TERMINAL  EXIT  FOR  LISP.

RECLAIM  [SUBR]

PACKJ  [SUBR]  $
PACKJ  IS  A  FUNCTION  OF  ONE  ARGUMENT  WHICH  IS  A  POINTER
TO  A  CHARACTER.  ITS  EFFECT  IS  TO  ADD  THIS  CHARACTER  TO  THE
TELETYPE  OUTPUT  BUFFER.  PACKJ  RETURNS  NIL.

BELL  [APVAL]  $
BELL  IS  FOR  THOSE  WHO  LIKE  A  LITTLE  AUDIO  OUTPUT  FROM
THEIR  PROGRAM.  BELL  IS  A  CONSTANT  THAT  EVALUATES  INTO  AN
ATOM  WITH  THE  PRINT  NAME  <BELL>.

EXPR;  FEXPR;  SUBR;  FSUBR;  APVAL;  PNAME;

FWCONS  [SUBR]  $
FWCONS  IS  A  FUNCTION  OF  ONE  ARGUMENT.  IT'S  ARGUMENT  IS
A  POINTER  TO  A  FULL  WORD.  FWCONS'S  VALUE  IS  A  POINTER  TO  A
NEW  FULL  WORD  WHICH  IS  A  COPY  OF  THE  ARGUMENT.

OBLIST  [APVAL]  4
AS  ON  THE  7090  THE  OBLIST  EVALUATES  INTO  ALL  ATOMIC
OBJECTS  THAT  HAVE  BEEN  INPUTED  INTO  THE  SYSTEM.  THE  OBLIST
ON  THE  235  HAS  ONLY  32  BUCKETS  INSTEAD  OF  THE  USUAL  64  BUT
IS  OTHERWISE  SIMILAR.

OPEN  [SUBR]  $
OPEN  IS  A  FUNCTION,  OF  ONE  OR  TWO  ARGUMENTS.  ITS
PURPOSE  IS  TO  DETERMINE  WHICH  FILE  READ,  ADVANCE,  AND
ADVANCE*  PULL  THEIR  DATA  FROM.  INITIALLY  THE  FILE  THAT  IS
OPEN  FOR  READING  IS  THE  USER'S  CURRENT  PROGRAM.  HOWEVER  IF
THE  USER  WISHES  TO  READ  ELSEWHERE  ON  THE  DISK  HE  JUST
SPECIFIES  THE  USER  NUMBER  AND  THE  FILE  NAME  TO  OPEN.  THE
FIRST  ARGUMENT  OF  OPEN  IS  THE  USER  NUMBER  AND  THE  SECOND  IS
THE  FILE  NAME.  FOR  EXAMPLE:  TO  READ  IN  TRACE-  FROM  THE
LIBRARY,  (OPEN('QUOTE  LIBEVA')  '(QUOTE  TRACE-')).  IT  SHOULD  BE
NOTED THAT THE USER NUMBER AND THE FILE NAME ARE BOTH ATOMS AND NEVER NUMBERS. THIS MEANS THAT AN USER NUMBER WITH ALL NUMERIC CHARACTERS MUST HAVE THE FIRST DIGIT QUOTED WITH A <SLASH> IN ORDER TO MAKE THE NUMBER LOOK LIKE AN ATOM. ALSO ANY FILE NAME WITH LESS THAN SIX CHARACTERS MUST BE FOLLOWED BY TRAILING BLANKS.

CONTINUED IN LSPFNE***
LSPFNE 21:22 SEPT. 7, 1966

100(DEFPROP SQR(LAMBDA(N)(PRJG(K X FLAG))(SETQ N(QUOTIENT(ADD1
110(SETQ X(PLUS N(SETQ FLAG 0.0))))2)LOORP(C3ND((GREATERP FLAG 12)
120(RETURN N)))(SETQ N(QUOTIENT(PLUS N(QUOTIENT X N))2))(SETQ FLAG(ADD1
130 FLAG))(G0 L30P)))EXPR)
140(LISTEN)
OPEN  [SUBR] $  (CONTINUED)
FOR EXAMPLE: TO OPEN FILE "ABCD" UNDER USER NUMBER 123456,
(OPENQUOTE /123456)(QUOTE ABCD/ / )). ONCE A FILE IS
OPENED READING COMMENCES WITH THE BEGINNING OF THE FILE WITH
THE USE OF A FILE READING COMMAND (READ, ADVANCE, ADVANCE*).
THE FIRST CHARACTER READ BY ADVANCE OR ADVANCE* WILL BE A
<CARRIAGE RETURN>. IF INSTEAD OF GIVING OPEN A USER NUMBER
AND A FILE NAME THE USER GIVES ONLY THE SINGLE ARGUMENT NIL,
THEN READING COMMENCES WITH THE BEGINNING OF THE USERS
CURRENT PROGRAM. THIS (OPEN()) ACTS AS A RESTORE.

INTERPRETQ  [SUBR] $
INTERPRETQ IS A FUNCTION DESIGNED TO GIVE EVAL BETTER
COMPATIBILITY WITH EVALQUOTE SYSTEMS. INTERPRETQ TAKES ONE
ARGUMENT. IF THIS ARGUMENT IS NIL THEN THE INTERPRETIVE
FUNCTION IS EVAL. HOWEVER IF THE ARGUMENT IS NOT NIL THEN
THE INTERPRETIVE FUNCTION BECOMES EVALQUOTE. THUS
(INTERPRETQ T) WILL CALL EVALQUOTE AND INTERPRETQNIL) WILL
BRING BACK EVAL. INTERPRETQ AFFECTS BOTH THE NORMAL
INTERPRETING FROM THE PROGRAM AND THE INTERPRETING DONE ON
STATEMENTS TYPED IN FROM A TYPewriter BY MEANS OF A LISTEN.
THE VALUE OF INTERPRETQ IS ITS PREVIOUS ARGUMENT. THIS IS
INITIALLY NIL SO THAT INTERPRETING STARTS WITH EVAL.

FUNCTION  [FSUBR]
MEMBER  [SUBR]
APPEND  [SUBR]
SUBST  [SUBR]

MAPLIST  [SUBR] $
MAPLIST'S DEFINITION HAS BEEN CHANGED TO:
(MAPLIST(LAMBDA(F,L)(COND
((NULL L)NIL)
((CONS(F L)(MAPLIST F(CDR L)))) ))EXPR)
THIS MEANS THAT THE ORDER OF MAPLISTS ARGUMENTS HAS BEEN
CHANGED FROM THE ORDER DESCRIBED IN THE LISP 1.5 PROGRAMMERS
MANUAL.

MAPCAR; MAP; MAPCON  [SUBR] $
LIKE MAPLIST, MAPCAR AND MAP AND MAPCON HAVE HAD THEIR
ORDER OF ARGUMENTS REVERSED FROM THEIR ORIgINAL DEFINITIONS.

COPY  [SUBR]
REVERSE  [SUBR]
LENGTH [SUBR]
CLOCK [SUBR] $
CLOCK$ IS A FUNCTION OF NO ARGUMENTS. ITS VALUE IS THE
NUMERICAL TIME OF DAY IN SIXTHS OF A SECOND.

LOGR; LOGAND; LOGXOR [FSUBR] $
THESE FUNCTIONS WORK NORMALLY EXCEPT THAT THEY ACCEPT
FLOATING POINT ARGUMENTS WITHOUT ERROR. IF GIVEN A FLOATING
POINT ARGUMENT THEY DO NOT FIX IT BUT USE ITS PATTERN OF
BITS DIRECTLY TO FORM THEIR RESULT. THE VALUE OF THESE
FUNCTIONS IS ALWAYS A FIXED POINT NUMBER.

ADD; SUB; [SUBR]
ZEREP; ONEP [SUBR]

LEFTSHIFT; RIGHTSHIFT [SUBR] $
The second argument of LEFTSHIFT OR RIGHTSHIFT (which is
the shift count) must be a fixed point number. These
functions shift the number specified as the first argument
(floating point argument is handled the same as in LOGR)
the specified number of bits. The effect on fixed point
numbers is multiplying or dividing by the appropriate power
of two. The forms (LEFTSHIFT A B) AND (RIGHTSHIFT A(MINUS
B)) ARE IDENTICAL EXCEPT IN THE CASE WHERE B IS ZER0. IN
THIS CASE DUE TO THE NATURE OF THE GE 235 A LEFTSHIFT 0
CAUSES A SHIFT OF THE SIGN BIT OF THE LOWER HALF INTO THE
UPPER HALF WHILE A RIGHTSHIFT 0 CAUSES A SHIFT OF THE SIGN
BIT FROM THE UPPER HALF INTO THE LOWER HALF. IN FIXED POINT
ARITHMETIC THE SIGN IN THE LOWER HALF IS IGNORED AND IS
FORCED TO AGREE WITH THE SIGN OF THE UPPER HALF. IN
FLOATING POINT ARITHMETIC THE SIGN OF THE LOWER HALF IS THE
SIGN OF THE NUMBER AND SIGN OF THE UPPER HALF IS THE SIGN OF
THE EXPONENT.

MAX; MIN [FSUBR]
RECIPE [SUBR]
ATTRIB [SUBR]
REMPREP [SUBR]
PR0P [SUBR]
QCNC [SUBR]
DEFINE

100 (DEFPRJP DEFLIST
110   (LAMBDA(L A)(MAPLIST
120     (FUNCTION(LAMBDA(X)(PUTPRJP(CAR X)(CADAR X)(CADR L))))
130     (CAR L)) )FEXPR)
140 (DEFLIST(150 (DEFINE(LAMBDA(L A)(EVAL
160   (LIST(QUOTE DEFLIST)(CAR L)(QUOTE EXPR))
170   A )) )FEXPR)
180 (LISTEN)

COND--

90 . USE EDIT RUNOFF TO LIST
100 CCOND HAS BEEN MODIFIED TO ACCEPT EXPRESSIONS OF THE FORM:
110
115 . RIG 66
120 (COND(P11,P12,...,PK)(P21,P22,...,P2L)...(PN1,PN2,...,PNM))
125 . RIG 60
130
140 CCOND EVALUATES ALL PII UNTIL IT FINDS THE FIRST THAT IS NOT NIL.
150 CCOND THEN EVALUATES THE REMAINDER OF PII, FOR ALL J, RETURNING
160 THE LAST PII FOR ITS VALUE. IF ALL PII ARE NIL, THEN THE VALUE OF
170 CCOND IS NIL. UNDER THIS DEFINITION IT IS POSSIBLE TO HAVE A
180 CONDITIONAL OF ONE EXPRESSION IN THE CASE OF THERE BEING ONLY
190 ONE PII. IN THIS CASE IF PII IS NOT NIL, THEN THE VALUE OF PII IS
200 THE VALUE OF CCOND. A LISP DEFINITION OF CCOND (HERE CALLED CCOND*) IS:
210
220  (DEFPRJP CCOND*
230  (LAMBDA(L A)(CCOND
240    ((NULL L)NIL)
250    (T(CND1(EVAL(CAR L)A)L A))
260    )) FEXPR)
270
280  (DEFPRJP CND1
290  (LAMBDA(E L A)(CCOND
300    ((NULL E)(CCOND
310      ((NULL(CDR L))NIL)
320      (T(COND1(EVAL(CADR L)A)(CDR L)A))
330      ))
340    (T(CCOND2 EXCDAR L)A))
350    )) EXPAR)
360
370  (DEFPRJP CCOND2
cond-- continued

380 (lambda (e l a) (cond
390    ((null l) e)
400    (t (cond2 (eval (car l) a) (cdr l) a))
410   )) expr)

base-

100 (defprop nat null expr)
110 (defprop and
120    (lambda (l a) (cond ((null l) t)
130        ((eval (car l) a)
140        ((eval (cons (quote and) (cdr l)) a))
150        (t ())))
160    ) expr)
170 (defprop or
180    (lambda (l a) (cond ((null l) ()
190        ((eval (car l) a) t)
200        (t (eval (cons (quote or) (cdr l)) a)))))
210 ) expr)

setq--

100 (defprop set
110    (lambda (l a) (cdr (rplacd (sassoc (eval (car l)) a)
120                                        a)
130    )
140    (function (lambda ()
150    (error (quote set)))))
160 ) expr)
170 (defprop setq
180    (lambda (l a) (cdr (rplacd (sassoc (car l) a)
190                                        a)
200    )
210    (function (lambda ()
220    (error (quote setq)))))
230 ) expr)
(DEFPR0P TH3REM(LAMBDA(S)(TH1 NIL NIL(CADDR S)(CADDR S))EXPR)
110.(DEFPR0P TH1(LAMBDA(A1 A2 A C)(C3ND((NULL A)(TH2 A1 A2 NIL NIL C)))
120.(T(OR(MEMBER(CAR A) C))(C3ND((AT3M(CAR A))(TH1(C3ND(MEMBER(CAR A))))
130(A1)A1)(T(CJNS(CAR A) A1)))A2(CDR A)C))(T(TH1 A1(C3ND(MEMBER(CAR A))))
140(A2)A2)T(CJNS(CAR A) A2)))CDR A)C))EXPR)
150.(DEFPR0P TH2(LAMBDA(A1 A2 C1 C2)(C3ND((NULL C)(TH A1 A2 C1 C2)))
160.(AT3M(CAR C))(TH2 A1 A2(C3ND(MEMBER(CAR C) C1)C1)(T(C3NS(CAR C)))
170(C1))C2(CDR C))(T(TH2 A1 A2 C1(C3ND(MEMBER(CAR C) C2)C2)(T(C3NS
180(CAR C) C2)(CDR C)EXPR)
190.(DEFPR0P TH3REM(LAMBDA(A1 A2 C1 C2)(C3ND((NULL A2)(AND(NOT(NULL C2))))
200.(TH1(CDR C)A1 A2 C1(CDR C)))(T(TH1(CDR C) A2(CDR C)1 C2)))EXPR)
210.(DEFPR0P TH1(LAMBDA(A1 A2 C1 C2)(C3ND((EQ(CAR U)(QUOTE N2)))
220.(TH1(CADR U)A1 A2 C1 C2))(EQ(CAR U)(QUOTE AND))TH1(CDR U))
230(A1 A2 C1 C2))EQ(CAR U)(QUOTE OR))AND(TH1(CADR U) A1 A2 C1 C2)
240.(TH1(CADR U)A1 A2 C1 C2))EQ(CAR U)(QUOTE IMPLIES)AND(TH1(CADR U)
250(CADR U)A1 A2 C1 C2))(EQ(CAR U)(QUOTE EQU))AND(TH1(CADR U))
260.(QUTE EQU))AND(TH1(CADR U) A1 A2 C1 C2)(TH2(CDR U) A1 A2 C1 C2))
270.(T(ERROR(LIST(QUOTE TH1)U A1 A2 C1 C2))EXPR)
280.(DEFPR0P TH1R(LAMBDA(A0 A2 C1 C2)(C3ND((EQ(CAR U)(QUOTE N2)))
290.(TH1R(CADR U)A0 A2 C1 C2))(EQ(CAR U)(QUOTE AND))TH1R(CDR U))
300(A0 A2 C1 C2))(TH1R(CADR U) A0 A2 C1 C2))EQ(CAR U)(QUOTE OR))AND(TH1R(CADR U)
310.(TH2R(CDR U)A0 A2 C1 C2))EQ(CAR U)(QUOTE IMPLIES)TH1R(CADR U)
320.(CADR U)A0 A2 C1 C2))EQ(CAR U)(QUOTE EQU)AND(TH1R(CADR U))
330.(CADR U)A0 A2 C1 C2)(TH1R(CADR U) C2))EQ(CAR U)(QUOTE IMPLIES)AND(TH1R(CADR U)
340.(LIST(QUOTE TH2R)U A0 A2 C1 C2))EXPR)
350.(DEFPR0P TH1R(LAMBDA(V1 V2 A1 A2 C1 C2)(C3ND((AT3M V1)(OR(MEMBER V1 C1))
360.(TH(C3NS V1 A1)A2 C1 C2))(T(OR(MEMBER V2 C2)(TH A1(C3NS V2 A2)C1 C2)))
370.))EXPR)
380.(DEFPR0P TH2R(LAMBDA(V A1 A2 C1 C2)(C3ND((AT3M V)(OR(MEMBER V A1))
390.(TH A1 A2(C3NS V1 C1)C2))(T(OR(MEMBER V2 C2)(TH A1 A2(C3NS V2 C2)))))
400.))EXPR)
410.(DEFPR0P TH2R(LAMBDA(V A1 A2 C1 C2)(C3ND((AT3M CAR V)(OR(MEMBER
420.(CAR V) C1)(TH1R(CADR V) C1NS(CAR V) A1)A2 C1 C2))(T(OR(MEMBER
430.(CAR V) C1)(TH1R(CADR V) A1 A2(C3NS(CAR V) C1 C2))(T(OR(MEMBER(CAR V
440.(A2)TH1R(CADR V) A1 A2(C3NS(CAR V) C2))EXPR)
450.(DEFPR0P TH2R(LAMBDA(V C1 V2 A1 A2 C1 C2)(C3ND((AT3M V1)(OR(MEMBER
460.(V1 C1)(TH1R V2(C3NS V1 A1)A2 C1 C2))(T(OR(MEMBER V1 C2)(TH1R
470.(V2 A1(C3NS V1) A2(C1 C2))EXPR)
480.(PR0G (A)A(SETQ ACERSETQCREATTY))(C3ND((NULL A)(G3 A))
490.(SETQ ACERSETQTHE0REM(CAR A))(C3ND((NULL A)(G3 A))
500.(G3 A))
510.(G3 A))
520.(G3 A))
530.(G3 A))
(DEFPRP BREAK (LAMBDA (FN WHEN WHAT) (PRG (TYPE DEF))
(COND ((SETQ DEF (GET FN (QUOTE EXPR))) (SETQ TYPE (QUOTE EXPR)))
((SETQ DEF (GET FN (QUOTE EXPR))) (SETQ TYPE (QUOTE EXPR)))
((PUTPRP FN (LIST (QUOTE LAMBDA) (QUOTE L A)) (LIST (QUOTE BREAK)))
(NIL (SETQ DEF (LIST FN (QUOTE UNDEFINED)) WHAT) (QUOTE EXPR))
(RETURN DEF))
((COND (EQ (CAR (CADDR DEF)) (QUOTE BREAK))) (RETURN (CONS FN (QUOTE BREAK)))
(CADDR DEF) (WHEN (LIST FN WHAT) TYPE))
(RETURN FN) (EXPR))
(DEFPRP BREAK (LAMBDA (L A) (PRG (*X))
(COND ((NULL (SETQ *X (ERSETQ (READTTY)))) (G0 B0))
((EQ (SETQ *X (CAR *X)) (QUOTE QUIT)) (ERROR (CADDR L)))
((EQ *X (QUOTE STJP)) (G0 B3))
((EQ *X (QUOTE RETURN)) (G0 B2))
((EQ *X (QUOTE EVAL))
((AND (SETQ *X (ERSETQ (QUOTE *)))))
(ERSETQ (PRINT (CAR *X)))
(TG B1))
(RETURN G0 B0))
(SETQ A (CONS (CONS (CAR (CADDR L)) (CAR *X)) A))
(G0 B1)
(B2 (COND (OR (NULL (SETQ *X (ERSETQ (READTTY))))
((NULL (SETQ *X (ERSETQ (CONS (CAR L) A))))) (G0 B0)))
(PRINT (CONS (CAR (CADDR L)) (QUOTE (EVALUATED))))
(SETQ A (CONS (CONS (CAR (CADDR L)) (CAR *X)) A))
(G0 B1)
(B3 (COND (EQ (CAR A) (CAR (CADDR L))) (SETQ *X (LIST (CDAR A))))
((NULL (SETQ *X (ERSETQ (CONS (CAR L) A)))) (G0 B0)))
(PRINT (APPEND (QUOTE (VALUE 0)) (CADDR L)))
((COND (NULL (ERSETQ (PRINT (CAR *X))))
(PRINT (QUOTE 0K)))
(RETURN (CAR *X))
EXPR)
(DEFPRP UNBREAK (LAMBDA (FN) (PRG (TYPE DEF))
(COND (SETQ DEF (GET FN (QUOTE EXPR))) (SETQ TYPE (QUOTE EXPR)))
((SETQ DEF (GET FN (QUOTE EXPR))) (SETQ TYPE (QUOTE EXPR)))
((RETURN (CONS FN (QUOTE (NOT BROKEN))))
(PUTPRP FN (LIST (QUOTE LAMBDA) (CADDR DEF) (CADDR (CADDR DEF) TYPE)))
(RETURN (CONS FN (QUOTE (NOT BROKEN))))
EXPR)
(DEFPRP BREAKLIST (LAMBDA (L A) (MAPCAR (QUOTE (LAMBDA (X) (BREAK X T NIL)))) L) FEXPR))
(DEFPR3P UNBREAKLIST(LAMBDA(L A)(MAPCAR(QUOTE UNBREAK)L))EXPR)

(DEFPR3P UNION
  (LAMBDA(X Y)(C3ND((NULL X)Y))
  (MEMBER(CAR X)Y)(UNION(CDR X)Y))
  (T(C3NS(CAR X)(UNION(CDR X)Y)))
  EXPR)

(DEFPR3P INTERSECTI0N
  (LAMBDA(X Y)(C3ND((NULL X)Y))
  (MEMBER(CAR X)Y)
  (C3NS(CAR X)(INTERSECTI0N(CDR X)Y))
  (T(INTERSECTI0N(CDR X)Y))
  EXPR)

(DEFPR3P REMVB
  (LAMBDA(L A)(PR3G(B C)A(C3ND((NULL L)(RETURN())))
  (SETQ B(CAR L))(SETQ C OB3LISTB(C3ND((NULL C)))(MEMBER B(CAR C))
  (PR3G()C3ND((EQ(CAR C)B)(RETURN(RPLACA C(CDAR C)))(SETQ C(CAR C))
  A(C3ND((EQ(CADR C)B)(RPLACD C(CDDR C)))(SETQ C(CDR C))(G3 A)))))
  (G3 B)))(SETQ L(CDR L)(G3 A))EXPR)

(DEFPR3P RENAME(LAMBDA(L A)(PR3G(B))(EVAL(LIST(QUOTE REMVB)(CAR L))NIL
  (SETQ B OB3LISTA(C3ND((NULL B)(RETURN())))(MEMBER(CADR L)(CAR B))
  (G3 B))(SETQ B(CDR B)(G3 A)B(SETQ B(REDUCE(CADR L)(CAR B)))
  (RPLACA B(CAR L))(RPLACACDR(REDUCE(QUOTE PNAME)(CDAR L)))(GET
  (CDAR L)(QUOTE PNAME))(RETURN(CADR L))EXPR)

(DEFPR3P REDUCE(LAMBDA(A L)(PR3G()B(C3ND((NULL)(RETURN()))
  (EQUAL(CAR L)A)(RETURN L))(SETQ L(CDR L)(G3 B))EXPR)

(REMVB)
LISP UPD

10 THE FOLLOWING IS A LIST OF CHANGES ADDED TO NEW LISP UPDATES.
20 THE CHANGES ARE LISTED IN REVERSE ORDER.
30 IN ORDER TO DIRECT NEW ADDITIONS AND CHANGES THE USER IS
40 ENCOURAGED TO APPEND QUESTIONS AND COMMENTS ONTO FILE "LISP??"
50 UNDER USER NUMBER 407886.

70 CURRENT VERSION: LISP 47
80 1) NUMBER NEW ROUNDS ON FLOATING POINT CONVERSION SO THAT (FIX 1.0)
90 NOW RETURNS 1 INSTEAD OF 0.
100 2) ON OUTPUT USING DOT NOTATION, THE DOT IS SURROUNDED BY A PAIR
110 OF BLANKS.
120 3) BECAUSE OF ABUSE OF ITS INTENT OPEN HAS BEEN UNDEFINED.
130 4) NEW FUNCTIONS: ATTRIB, CONC, PROG, REMPREE, MAPCON (PLEASE
140 NOTE THAT LIKE MAPLIST AND MAPCAR THE ARGUMENTS OF MAPCON HAVE BEEN
150 REVERSED.)
160 5) THE NAME OF THE GARBAGE COLLECTOR FUNCTION HAS BEEN CHANGED TO
170 RECLAIM AS STATED IN THE "LISP 1.5 PROGRAMMERS MANUAL."
6) The size of the output buffer has been decreased causing more
swaps with programs that have almost all output but giving more free
storage.

180 6) The size of the output buffer has been decreased causing more
swaps with programs that have almost all output but giving more free
storage.

200 LISP 46

200 1) A previous bug in numberp has been fixed
210 2) The form if numbers has been redefined so that numbers look
220 like atoms whose first property is fixnum or flonum but instead
230 of having a further property list after the number property
240 it points directly to the number. In other words (cddr number) is
250 a pointer into full-word space to a cell that contains the value
260 of the number.

280 3) Storage has been reallocated so that there is less full-word
290 space and less push-down list but more free-storage.

300 4) New protection features have been added so that it is more
difficult (maybe even impossible) for the user to damage time-
sharing.

320 5) The representation of constants in property lists has been
changed. No longer are constants depressed a level in the list
structure after the apval property flag. This means that cset
is defined as (prog2(putr3p arg1 arg2(quote apval)arg2) instead
of (prog2(putr3p arg1(list arg2)(quote apval))arg2) as previously.

400 6) The size of buffer has been changed from 81 characters to
only 40 characters. The reason for this is that buffer can only
be used for numbers and characters are illegal.

430 LISP 45

450 1) New functions: clock, logor, logand, logxor, leftshift,
rightshift, add1, sub1, onep, zerop, max, min, recip, length.

470 2) The flat phase of the arithmetic functions has been fixed so
that it floats a fixed point zero correctly.

490 3) New error--error i4. Error i4 is given by an arithmetic
function that finds a floating point argument when it was expecting
a fixed point argument. Leftshift and rightshift give this error
if a floating point number is given as their shift count.

530 LISP 44

550 1) A bucket sorted @blist has been added. The @blist consists
of 32 sublist (not the usual 64) in order that an atom
only a small fraction of the @blist need be scanned. The result
is that read and readtty have been speeded up by almost a factor
of five over previous versions.

600 2) cond has been redefined to take a greater variety of forms
of arguments. See cond-*** for details.

620 3) The functions maplist, mapcar, and map have been added. Their
definitions have been changed slightly so that their first
argument is a function and their second argument is a list to apply
this function to. Normal order is to have the first argument
be the list and the function second but this seems awkward consider-
the mathematical notation of f(x) for functions.
4) EVAL AND APPLY HAVE BEEN MODIFIED SO THAT THEY DO NOT MAKE A
SPECIAL CHECK FOR NIL ANY LONGER. THE RESULT IS THAT
(APPLY NIL ARGS ALIST) GIVES UNDEFINED FUNCTION--NIL.
HOWEVER, (EVAL NIL ALIST) STILL RESULTS IN NIL, SINCE NIL IS
A CONSTANT THAT EVALUATES INTO ITSELF.
OTHER CHANGES WERE MADE TO IMPROVE THE SPEED OF LISP
(LIKE ALL INTERPRETIVE SYSTEMS LISP IS INTOLERABLY SLOW).
NONE OF THESE CHANGES CAN BE NOTICED BY THE USER.
LISP HAS BEEN GIVEN A NEW ERROR--ERROR TRA. THIS ERROR IS
GIVEN IF THE USER ATTEMPTS TO MAKE A TRANSFER INTO MEMORY USING
SUBR OR FSUBR FLAGS. ONLY PREDEFINED FUNCTIONS ARE ALLOWED TO
BE SUBRS OR FSUBRS.
THE FOLLOWING IS A CATALOG OF LISP PROGRAMS. THE LISP PROGRAMS IN THE LIBRARY MAY NOT HAVE ALL THE NEEDED FUNCTIONS INCLUDED WITHIN THEM BUT USUALLY ONE WILL FIND THE NECESSARY FUNCTIONS ELSEWHERE IN THE LIBRARY.

NOTE: THOSE FILES THAT CONTAIN SAMPLE DEFINITIONS WILL NOT RUN ON THE LISP SYSTEM. SINCE THESE FUNCTIONS ARE PREDEFINED A MEMORY ERROR WILL OCCUR IF THEY ARE REDEFINED. THESE DEFINITIONS ARE ONLY USE TO GIVE THE USER AN IDEA ON HOW THE SYSTEM WORKS.

ANYONE HAVING PROGRAMS OF INTEREST TO LISP USERS AND THINKS THEY SHOULD BE IN THE LIBRARY SHOULD CONTACT STEVE HOBBS AT THE COMPUTATION CENTER.

LISP02: UNION; INTERSECTION;

DEFINE: DEFLIST; DEFINE

SETQ--: CONTAINS SAMPLE DEFINITIONS OF SET AND SETQ.

TRACE--: CONTAINS A SIMULATOR FOR THE TRACE FEATURE IN 7090 LISP.

BOTH TRACE AND UNTRACE ARE FUNCTIONS THAT TAKE AN UNLIMITED NUMBER OF ARGUMENTS WHICH ARE AUTOMATICALLY QUOTED. FOR EXAMPLE: TRACE ALPHA, BETA AND GAMMA (TRACE ALPHA BETA GAMMA). TRACE AND UNTRACE WILL NOT WORK ON PREDEFINED FUNCTIONS.


BOOLE--: CONTAINS A SAMPLE DEFINITION OF LISP BOOLEAN FUNCTIONS: AND; OR; NOT;

LISP-1 AND LISP-2: CONTAINS A DISCUSSION OF DIFFERENCES BETWEEN THE DARTMOUTH VERSION OF LISP AND THE LISP DESCRIBED IN THE "LISP 1.5 PROGRAMMERS MANUAL."

LSPFNC, LSPFND AND LSPFNE: CONTAINS A LIST OF PREDEFINED ATOMS AND HOW THEY DIFFER FROM THE "LISP 1.5 PROGRAMMERS MANUAL."

LSPERR: DESCRIBES THE POSSIBLE LISP ERROR MESSAGES.

TRCSET; UNTTRACESET; TRACESET; TRACES SETQ IN FUNCTIONS. (DO NOT USE TRACSET AFTER A TRACE AS ONLY THE TRACE ROUTINE WILL BE TRACESET. IF ONE WANTS TO USE BOTH TRACE AND TRACSET ON THE SAME FUNCTION THEN TRACSET IT FIRST FOLLOWED BY TRACE.)
LIBEVA CONTINUED

590 LSPUPD: CONTAINS A LIST CHANGES OR ADDITIONS THAT ARE MADE
600 IN EACH CHANGE IN LISP.
610
620 REM33: REMJB; RENAME
630
640 MKNAM -: CONTAINS A SPECIAL VERSION OF MKNAM. IT READS CONSECUTIVE
650 CHARACTERS FROM THE TTY UNTIL IT FINDS A BREAK CHARACTER (DEFINED
660 AS VARIABLE BRK) AND THEN RETURNS THE ATOM WITH THIS PRINT NAME.
670 THE BREAK CHARACTER THAT MARKED THE END OF THE ATOM CAN BE FOUND
680 AS VARIABLE LCHAR. IT SHOULD BE NOTED THAT THIS PROGRAM DOES
690 NOT HANDLE NUMBERS BUT ONE MUST INSTEAD USE PACK AND NUMAB.
LSPERR*-----OCTOBER 30, 1968

THIS FILE IS A LIST OF POSSIBLE ERROR MESSAGES UNDER THE DARTMOUTH VERSION OF LISP. ALL MESSAGES GIVEN BY LISP THAT AGREE WITH THE "LISP 1.5 PROGRAMMERS MANUAL" HAVE IDENTICAL MEANING; HOWEVER, SOME ERRORS HAVE BEEN ADDED AND SOME HAVE BEEN REMOVED.

ERROR A1: APPLIED FUNCTION CALLED ERROR. THIS IS AN ERROR FORCED BY THE PROGRAMMER BY THE EVALUATION OF THE FUNCTION ERROR. ERROR IS A FUNCTION OF ONE ARGUMENT, ON THE LINE AFTER THE ERROR TYPE THE EVALUATION OF ERROR'S ARGUMENT IS PRINTED. ERROR IS USEFUL AS A DEBUGGING AID.

ERROR A2: FUNCTION OBJECT HAS NO DEFINITION--APPLY. APPLY HAS BEEN ASKED TO EVALUATE AN UNDEFINED FUNCTION. AFTER THE ERROR TYPE THE FUNCTION THAT WAS UNDEFINED IS PRINTED.

ERROR A4: SETQ GIVEN ON A NONEXISTENT PROGRAM VARIABLE. THE NONEXISTENT VARIABLE IS PRINTED AFTER THE ERROR TYPE.

ERROR A5: SET GIVEN ON A NONEXISTENT PROGRAM VARIABLE. THE NONEXISTENT VARIABLE IS PRINTED AFTER THE ERROR TYPE.

ERROR A6: Go (OR Go*) REFERS TO AN UNLABELLED POINT. THE UNLABELLED POINT IS PRINTED AFTER THE ERROR TYPE.

ERROR A8: EVAL ASKED TO EVALUATE AN UNBOUND VARIABLE. THE VARIABLE THAT IS NOT DEFINED (BY LAMBDA, PROG, OR CSET) IS PRINTED. THIS ERROR OFTEN OCCURS WHEN THERE IS A PARENTHESES MISMATCH.

ERROR A9: FUNCTION OBJECT HAS NO DEFINITION--EVAL. THE UNDEFINED FUNCTION IS PRINTED. THIS ERROR OFTEN OCCURS WHEN THERE IS A PARENTHESES MISMATCH.

ERROR A10: PUTPROP'S FIRST ARGUMENT IS NOT ATOMIC. THIS ERROR OCCURS WITH DEFPROP; CSET AND CSETQ.

ERROR CH1: TOO MANY CHARACTERS--PACK. PACK HAS BEEN CALLED TO PACK MORE 81 CHARACTERS INTO BUFFER WITHOUT A CLEARBUFFER.

ERROR CH2: FLOATING POINT NUMBER OUT RANGE--NUMOB. IT IS POSSIBLE ALSO TO GET THIS ERROR ON DECIMAL INTEGERS OF MORE THAN 12 CHARACTERS SINCE AT THIS POINT NUMBO CANNOT HAVE DECIDED WHETHER OR NOT THE NUMBER IS FLOATING POINT.

ERROR CH4: BAD CHARACTER--NUMOB

ERROR DSK: THERE HAS BEEN A DISK ERROR. IF THIS ERROR OCCURS DURING AN OPEN THEN READ, ADVANCE AND ADVANCE* BECOME UNDEFINED.

ERROR F2: PAIR'S FIRST ARGUMENT LIST IS TOO SHORT. THE EXTRA ELEMENTS ARE PRINTED AFTER THE ERROR TYPE. THIS MOST OFTEN HAPPENS WHEN TOO MANY ARGUMENTS ARE GIVEN TO A LAMBDA BOUND FUNCTION.
ERROR F3: PAIR'S SECOND ARGUMENT LIST IS TOO SHORT. THE EXTRA ELEMENTS ARE PRINTED AFTER THE ERROR TYPE. THIS MOST OFTEN HAPPENS WHEN TOO FEW ARGUMENTS ARE GIVEN TO A LAMBDA BOUND FUNCTION.

ERROR G1: ARITHMETIC OVERFLOW OR DIVIDE CHECK. ON THE CE 235 BOTH FIXED AND FLOATING POINT OPERATIONS CAN BE TRAPPED.

ERROR G2: OUT OF PUSH-DOWN LIST. THIS HAPPENS WHEN RECURSION IS TOO DEEP. IT OFTEN INDICATES INFINITE RECURSION.

ERROR GC2: NO WORDS COLLECTED--GARbage COLLECTER. THIS MEANS THE EVALUATION OF PRESENT FUNCTIONS REQUIRE MORE THAN THE MACHINE'S CAPACITY FOR LIST STRUCTURE. THE 235 HAS A VERY SMALL CAPACITY.

ERROR I3: ARGUMENT OF A NUMERIC FUNCTION IS NOT A NUMBER.

ERROR I4: ARGUMENT OF A FUNCTION EXPECTING A FIXED POINT NUMBER IS A FLOATING POINT NUMBER.

ERROR MEM: THERE HAS BEEN AN ATTEMPT TO MODIFY A CELL THAT IS NOT IN FREE-STORAGE OR FULL-WORD-SPACE. THIS MAY HAPPEN IF THE USER TRIES TO MODIFY A PREDEFINED ATOM.

ERROR P2: ATOM HAS NO PRINT NAME--PRIN1. THIS HAPPENS WHEN AN ATOM IS ATTEMPTED TO BE PRINTED WITHOUT A PNAME, FIXNUM OR FLONUM ON ITS PROPERTY LIST. THIS MOST OFTEN HAPPENS WHEN ONE TRIES TO PRINT CARS OR CDRS BEYOND THE ATOMIC LEVEL.

ERROR P3: BAD BASE--SETOBASE OF SETIBASE. ONLY NUMBER BASES BETWEEN TWO AND TEN CAN BE USED FOR INTEGER INPUT-OUTPUT.

ERROR R1: FIRST OBJECT ON THE INPUT LIST IS ILLEGAL--READ. THIS MOST OFTEN HAPPENS WHEN THERE IS A MISPLACED <DOT> OR <RIGHT PARENTHESIS>.

ERROR R2: CONTEXT ERROR WITH DOT NOTATION--READ.

ERROR R3: ILLEGAL CHARACTER--READ. THE INPUT OF BAD END OF FILE. THIS IS CAUSED WHEN READ, ADVANCE OR ADVANCE* ATTEMPT TO READ BEYOND THE END OF A FILE.

ERROR R7: FILE NOT FOUND--OPEN. AN ATTEMPT WAS MADE TO OPEN A NONEXISTENT FILE. IF OPEN IS UNSUCCESSFUL THEN READ, ADVANCE AND ADVANCE* ARE UNDEFINED.

ERROR TRA: THIS ERROR OCCURS WHEN THE INTERPRETER DISCOVERS XTW. THIS PREVENTS THE USER FROM TRANSFERRING TO ILLEGAL LOCATIONS IN MEMORY.

END