```
 TYPE PL10C
** PROLOG TO DEC 10 MACHINE CODE COMPILER, VERSION 13 SEP 1975.
** ------------------------------------------------------------.

-LIREFICHIER.


** OPERATOR DECLARATIONS.

+;(DG,1).
+WHERE(DG,2).
+OR(DG,3).    +ELSE(DG,3).
+&(DG,4).
+=(DG,5).    +<(DG,5).    +>(DG,5).    +LE(DG,5).    +GE(DG,5).
+IS(DG,5).   +BECOMES(DG,5).
+:(DG,6).    +.(DG,6).
+!(DG,7).    ++(GD,7).    +-(GD,7).
+#(GD,8).    +/(GD,8).
+MOD(GD,9). +@(GD,9).

** A FAIRLY GOOD SOLUTION TO THE TASK OF EVALUATING INTEGER
** EXPRESSIONS USING A FIXED NUMBER (IN THIS CASE 7) OF
** FAST ACCUMULATORS IS EXPRESSED BY:-
**      INTEVAL(*X,*I,*J,*K,*N,*RESULT, *CODE )   WHERE
** INSTRUCTIONS *CODE EVALUATE EXPRESSION *X TO PRODUCE A VALUE
** WHICH IS EITHER THE CONTENTS OF AC(*I) OR IS A LITERAL VALUE.
** ACCUMULATORS AC(*J) TO AC(*K-1) AND MEMORY LOCATIONS MEM(*N)
** AND ABOVE MAY BE SIDE-EFFECTED.
** NOTE THAT AC(*I1)=AC(*I2) IF *I1=*I2 MOD 7
**                 WHERE 7 IS THE NUMBER OF FAST ACCUMULATORS.


+INTEVAL(QQQ(*Z,*EITHER),
         *I,*I,*I,*N,LIT(0),  JRST(ERRCG) )

   -VAR(*Z)-/-ERROR("VARIABLE UNASSIGNED").

+INTEVAL(QQQ(*M,MULTIPLE),
         *I,*I,*I1,*N,AC(*I), MOVE(R2,X(*M));
                              HLRZ(R1,R2);
                              CAIE(R1,INT);
                              JSP(C,INTLD);
                              HRRZ(AC(*I),R2) )

   -/-PLUS(*I,1,*I1).

+INTEVAL(*X,
         *I,*I,*I,*N,LIT(*X), VOID )

   -UNIV(*X,(*L.*R).NIL)
   -CHIFFRE(*L)-/.

** SUB-EXPR-2:                          !-----!=!-------------!
**                                     *J2   *I2             *K2
**
** SUB-EXPR-1:       !------!=!-------------!
**                 *J1     *I1             *K1
**
** OPERATION:              !---!
**                        *J3 *K3
**
```

Comp-2

```
** EXPR:              !-------!=!-------------------------------!
**                   *J      *I                                *K
**.

+INTEVAL(*X,
         *I,*J,*K,*N,*RESULT, *ARG2;
                              *SAVE;
                              *ARG1;
                              *OP )

   -NORMINTEXPR(*X,*F,*X1,*X2)-/
   -PLUS(*N,1,*N1)
   -INTEVAL(*X1,*I,*J1,*K1A,*N1,*RESULT1A, *ARG1A )
   -INTEVALSTEP1(*RESULT1A,*RESULT1,
                 *ARG1A,*ARG1,
                 *K1A,*K1,   *F,*I)
   -INTEVAL(*X2,*K1,*J2,*K2,*N,*RESULT2A, *ARG2 )
   -INTEVALSTEP2(*RESULT2A,*RESULT2,
                 *SAVE,*J1,*K1,*N)
   -INTOP(*F,*RESULT1,*RESULT2,*RESULT,*I, *OP )
   -AFFECTS(*F,*EXTENTOP)-PLUS(*I,*EXTENTOP,*K3)
   -MAX(*K2,*K3,*K)-MIN(*J1,*J2,*J).

+INTEVAL(*Y,
         *I,*J,*K,*N,*RESULT, *ARG;
                              *OP )

   -UNARYEXPR(*Y,*F,*X)-/
   -INTEVAL(*X,*I,*J,*K1,*N,*RESULT1, *ARG )
   -UNARYOP(*F,*RESULT1,*RESULT,*I, *OP )
   -PLUS(*I,1,*K2)
   -MAX(*K1,*K2,*K).

+INTEVAL(*X,
         *I,*I,*I1,*N,AC(*I), MOVEI(AC(*I),*X) )

   -KNOWNADDR(*X)-/
   -PLUS(*I,1,*I1).

+INTEVAL(*X,
         *I,*I,*I,*N,LIT(0),  JRST(ERRCG) )

   -ERROR("INTEGER EXPR REQUIRED").

+INTEVALSTEP1(LIT(*Z),AC(*I),
               *ARG1A, *ARG1A;MOVEI(AC(*I),*Z) ,
              *K1A,*K1, *F,*I)
   -NOREVERSEOP(*F)-/
   -PLUS(*K1A,1,*K1).

+INTEVALSTEP1(*RESULT1,*RESULT1,
               *ARG1,*ARG1,
               *K1,*K1, *F,*I).

+INTEVALSTEP2(AC(*I2),MEM(*N),
               MOVEM(AC(*I2),MEM(*N)) ,*J1,*K1,*N)
   -MOINS(*K1,*J1,*EXTENT)
   -INF(6,*EXTENT)-/.

+INTEVALSTEP2(*RESULT2,*RESULT2,

               VOID ,*J1,*K1,*N).
```

```
+KNOWNADDR(HEAP).
+KNOWNADDR(FNTAB).

+INTREL(*X(*Y).
+INTREL(*X)*Y).
+INTREL(*X LE *Y).
+INTREL(*X GE *Y).

+NORMINTEXPR(*X+*Y,ADD,*X,*Y).
+NORMINTEXPR(*X-*Y,SUBTRACT,*X,*Y).
+NORMINTEXPR(*X*Y,MULTIPLY,*X,*Y).
+NORMINTEXPR(*X/*Y,DIVIDE,*X,*Y).
+NORMINTEXPR(*X(*Y,LESSTHAN,*X,*Y).
+NORMINTEXPR(*X)*Y,LESSTHAN,*Y,*X).
+NORMINTEXPR(*X LE *Y,LESSOREQ,*X,*Y).
+NORMINTEXPR(*X GE *Y,LESSOREQ,*Y,*X).
+NORMINTEXPR(WD(*Y) BECOMES *X,ASSIGN,*X,*Y).
+NORMINTEXPR(LH(*Y) BECOMES *X,LASSIGN,*X,*Y).
+NORMINTEXPR(RH(*Y) BECOMES *X,RASSIGN,*X,*Y).
+NORMINTEXPR(XWD(*L,*R),XWD,*R,*L).

+NOREVERSEOP(SUBTRACT).
+NOREVERSEOP(DIVIDE).
+NOREVERSEOP(ASSIGN).
+NOREVERSEOP(LASSIGN).
+NOREVERSEOP(RASSIGN).
+NOREVERSEOP(XWD).

+AFFECTS(ADD,1).
+AFFECTS(SUBTRACT,1).
+AFFECTS(MULTIPLY,1).
+AFFECTS(DIVIDE,2).
+AFFECTS(ASSIGN,1).
+AFFECTS(LASSIGN,1).
+AFFECTS(RASSIGN,1).
+AFFECTS(LESSTHAN,0).
+AFFECTS(LESSOREQ,0).
+AFFECTS(XWD,1).

+INTOP(ADD,      AC(*I),  AC(*I1), AC(*I), *I,ADD(AC(*I),AC(*I1)) ).
+INTOP(ADD,      AC(*I),  MEM(*N), AC(*I), *I,ADD(AC(*I),MEM(*N)) ).
+INTOP(ADD,      AC(*I),  LIT(*N), AC(*I), *I,ADDI(AC(*I),*N) ).
+INTOP(ADD,      LIT(*N), AC(*I),  AC(*I), *I,ADDI(AC(*I),*N) ).
+INTOP(ADD,      LIT(*N1),LIT(*N2),LIT(*N),*I,VOID )
   -PLUS(*N1,*N2,*N).
+INTOP(SUBTRACT,AC(*I),  AC(*I1), AC(*I), *I,SUB(AC(*I),AC(*I1)) ).
+INTOP(SUBTRACT,AC(*I),  MEM(*N), AC(*I), *I,SUB(AC(*I),MEM(*N)) ).
+INTOP(SUBTRACT,AC(*I),  LIT(*N), AC(*I), *I,SUBI(AC(*I),*N) ).
+INTOP(SUBTRACT,LIT(*N1),LIT(*N2),LIT(*N),*I,VOID )
   -MOINS(*N1,*N2,*N).
+INTOP(MULTIPLY,AC(*I),  AC(*I1), AC(*I), *I,IMUL(AC(*I),AC(*I1)) ).
+INTOP(MULTIPLY,AC(*I),  MEM(*N), AC(*I), *I,IMUL(AC(*I),MEM(*N)) ).
+INTOP(MULTIPLY,AC(*I),  LIT(*N), AC(*I), *I,IMULI(AC(*I),*N) ).
+INTOP(MULTIPLY,LIT(*N), AC(*I),  AC(*I), *I,IMULI(AC(*I),*N) ).
+INTOP(MULTIPLY,LIT(*N1),LIT(*N2),LIT(*N),*I,VOID )
   -MULT(*N1,*N2,*N3).
+INTOP(DIVIDE,  AC(*I),  AC(*I1), AC(*I), *I,IDIV(AC(*I),AC(*I1)) ).
+INTOP(DIVIDE,  AC(*I),  MEM(*N), AC(*I), *I,IDIV(AC(*I),MEM(*N)) ).
```

```
+INTOP(DIVIDE,  AC(*I),  LIT(*N), AC(*I), *I,IDIVI(AC(*I),*N) ).
+INTOP(DIVIDE,  LIT(*N1),LIT(*N2),LIT(*N),*I,VOID )
   -DIV(*N1,*N2,*N).
+INTOP(ASSIGN,  AC(*I),  AC(*I1), AC(*I), *I,MOVEM(AC(*I),
                                                  0:AC(*I1)) ).
+INTOP(ASSIGN,  AC(*I),  MEM(*N), AC(*I), *I,MOVE(ACH,MEM(*N));
                                                  MOVEM(AC(*I),ACH(0)) ).
+INTOP(ASSIGN,  AC(*I),  LIT(*N), AC(*I), *I,MOVEM(AC(*I),*N) ).
+INTOP(ASSIGN,  LIT(*N1),LIT(*N2),AC(*I), *I,MOVEI(AC(*I),*N1);
                                                  MOVEM(AC(*I),*N2) ).
+INTOP(LASSIGN, AC(*I),  AC(*I1), AC(*I), *I,HRLM(AC(*I),
                                                  0:AC(*I1)) ).
+INTOP(LASSIGN, AC(*I),  MEM(*N), AC(*I), *I,MOVE(ACH,MEM(*N));
                                                  HRLM(AC(*I),ACH(0)) ).
+INTOP(LASSIGN, AC(*I),  LIT(*N), AC(*I), *I,HRLM(AC(*I),*N) ).
+INTOP(LASSIGN, LIT(*N1),LIT(*N2),AC(*I), *I,MOVEI(AC(*I),*N1);
                                                  HRLM(AC(*I),*N2) ).
+INTOP(RASSIGN, AC(*I),  AC(*I1), AC(*I), *I,HRRM(AC(*I),
                                                  0:AC(*I1)) ).
+INTOP(RASSIGN, AC(*I),  MEM(*N), AC(*I), *I,MOVE(ACH,MEM(*N));
                                                  HRRM(AC(*I),ACH(0)) ).
+INTOP(RASSIGN, AC(*I),  LIT(*N), AC(*I), *I,HRRM(AC(*I),*N) ).
+INTOP(RASSIGN, LIT(*N1),LIT(*N2),AC(*I), *I,MOVEI(AC(*I),*N1);
                                                  HRRM(AC(*I),*N2) ).
+INTOP(LESSTHAN,AC(*I),  AC(*I1), VOID,   *I,CAML(AC(*I),AC(*I1));
                                                  JRST(EPFAIL) ).
+INTOP(LESSTHAN,AC(*I),  MEM(*N), VOID,   *I,CAML(AC(*I),MEM(*N));
                                                  JRST(EPFAIL) ).
+INTOP(LESSTHAN,AC(*I),  LIT(*N), VOID,   *I,CAIL(AC(*I),*N);
                                                  JRST(EPFAIL) ).
+INTOP(LESSTHAN,LIT(*N), AC(*I),  VOID,   *I,CAIG(AC(*I),*N);
                                                  JRST(EPFAIL) ).
+INTOP(LESSTHAN,LIT(*N1),LIT(*N2),VOID,   *I,*CODE )
   -DO( *N2>*N1 & *CODE=VOID ELSE *CODE=JRST(EPFAIL) ).
+INTOP(LESSOREQ,AC(*I),  AC(*I1), VOID,   *I,CAMLE(AC(*I),AC(*I1));
                                                  JRST(EPFAIL) ).
+INTOP(LESSOREQ,AC(*I),  MEM(*N), VOID,   *I,CAMLE(AC(*I),MEM(*N));
                                                  JRST(EPFAIL) ).
+INTOP(LESSOREQ,AC(*I),  LIT(*N), VOID,   *I,CAILE(AC(*I),*N);
                                                  JRST(EPFAIL) ).
+INTOP(LESSOREQ,LIT(*N), AC(*I),  VOID,   *I,CAIGE(AC(*I),*N);
                                                  JRST(EPFAIL) ).
+INTOP(LESSOREQ,LIT(*N1),LIT(*N2),VOID,   *I,*CODE )
   -DO( *N1>*N2 & *CODE=JRST(EPFAIL) ELSE *CODE=VOID ).
+INTOP(XWD,     AC(*I),  AC(*I1), AC(*I), *I,HRL(AC(*I),AC(*I1)) ).
+INTOP(XWD,     AC(*I),  MEM(*N),  AC(*I), *I,HRL(AC(*I),MEM(*N)) ).
+INTOP(XWD,     AC(*I),  LIT(*N),  AC(*I), *I,HRLI(AC(*I),*N) ).
+INTOP(XWD,     LIT(*N1),LIT(*N2),AC(*I), *I,MOVEI(AC(*I),*N1);
                                                  HRLI(AC(*I),*N2) ).

+UNARYEXPR(WD(*X),WORD,*X).
+UNARYEXPR(LH(*X),LHALF,*X).
+UNARYEXPR(RH,RHALF,*X).

+UNARYOP(WORD, AC(*I), AC(*I),*I, MOVE(AC(*I),0:AC(*I)) ).
+UNARYOP(WORD, LIT(*N),AC(*I),*I, MOVE(AC(*I),*N) ).
+UNARYOP(LHALF,AC(*I), AC(*I),*I, HLRZ(AC(*I),0:AC(*I)) ).
+UNARYOP(LHALF,LIT(*N),AC(*I),*I, HLRZ(AC(*I),*N) ).
+UNARYOP(RHALF,AC(*I), AC(*I),*I, HRRZ(AC(*I),0:AC(*I)) ).
+UNARYOP(RHALF,LIT(*N),AC(*I),*I, HRRZ(AC(*I),*N) ).
```

```
** COMPILATION OF A SINGLE CLAUSE.


+CLAUSE(*C,
        *F,*M,*ARG1,            *ALLOCATE;
                                *UNIFY1;
                                *UNIFY;
                                *SAVEFL;
                                *BODY;
                                JRST(A(*M));
                                *DATA1;
                                *DATA2;
                                *DATA3;
                                *DATA4 )

   -COMPLETERHS(*C,*P WHERE *Q)
   -MARKVARS(*P WHERE *Q)
   -CHECKSLASH(*P,*P1,*SAVEFL)
   -UNIV(*P1,*F.*A)-LENGTH(*A,*M)
   -HDNTL(*A,*A0,*A1)
   -LTERMFIRST(*A0,*ARG1,*N1,*UNIFY1,*DATA1)
   -LTERMSOUTER(*A1,*N1,*N2,1, *UNIFY , *DATA2 )
   -BODY(*Q,*N2,*N3, *BODY , *DATA3 )
   -MOINS(*N3,2,*N)
   -ALLOCATE(*N, *ALLOCATE , *DATA4 ).


+HDNTL(NIL,VOID,NIL).
+HDNTL(*X.*Y,*X,*Y).


** THE BODY (RIGHT-HAND SIDE) OF THE CLAUSE.


+BODY(TRUE,*N,*N, VOID , VOID )-/.

+BODY(*Q,
      *NO,*N,                   *INITIALISE;
                                MOVE(X,V);
                                *GOAL;
                                MOVE(A,X(1));
                                HLRZ(X,A) ,

                                *DATA )

   -GOALS(*Q,*NO,*N, *GOAL , *DATA )
   -INITIALISE(*NO,*N, *INITIALISE ).


** ALLOCATION OF SPACE ON THE STACK FOR THE CLAUSE'S VARIABLES.


+ALLOCATE(0,                    VOID ,

                                VOID )-/.
```

```
+ALLOCATE(1,                    AOBJP(ST,STOFL),

                                VOID )-/.

+ALLOCATE(2,                    AOBJP(ST,STOFL);
                                AOBJP(ST,STOFL) ,

                                VOID )-/.

+ALLOCATE(*N,                   ADD(ST,DBL(*N));
                                JUMPGE(ST,STOFL) ,

                                VOID )

   -INF(*N,10)-/.

+ALLOCATE(*N,                   ADD(ST,*L);
                                JUMPGE(ST,STOFL) ,

                              *L:;
                                XWD(*N,*N) ).


** EVALUATION OF THE GOALS COMPRISING THE BODY OF THE CLAUSE.


+GOALS(*Q1 OR *Q2,
      *NO,*N,                   PUSH(ST,V);
                                MOVEI(V,ST(0));
                                AOBJP(ST,STOFL);
                                HRLZM(X,V(1));
                                *DISJUNCT;
                              *EXIT: ,

                                *DATA )

   -/-DISJUNCT(*Q1 OR *Q2,*NO,*N,*EXIT, *DISJUNCT , *DATA ).

+GOALS(*Q1&*Q2,
      *NO,*N,                   *GOAL1;
                                *GOAL2 ,

                                *DATA )

   -/
   -GOALS(*Q1,*NO,*N1, *GOAL1 , *DATA1 )
   -GOALS(*Q2,*N1,*N,  *GOAL2 , *DATA2 )
   -SHORTEN( *DATA1;*DATA2 , *DATA ).

+GOALS(*Q!,
      *NO,*N,                   *GOAL;
                                HRRZ(R1,X(0));
                                MOVEM(R1,V(0)) ,

                                *DATA )

   -/
   -GOALS(*Q,*NO,*N, *GOAL , *DATA ).
```

```
+GOALS(*X,*NO,*N, *CODE , *DATA )
   -NOTRACE-/-GOAL(*X,*NO,*N, *CODE , *DATA ).

+GOALS(*X,
       *NO,*N,                  JSP(A,PRED(SORTE));
                                *ARG;
                                OUTSTR(CRLF);
                                *CODE ,

                                *DATA )

   -RTERM(*X,OUTER,*NO,*N, *ARG , *DATA1 )
   -GOAL(*X,*N,*N1, *CODE , *DATA2 )
   -SHORTEN( *DATA1;*DATA2 , *DATA ).



+GOAL(*Y IS *X,
      *NO,*N,                   *ARG;
                                *RESULT ,

                                VOID )

   -/
   -INTLOAD(*X, *ARG )
   -INTRESULT(*Y,*NO,*N, *RESULT ).

+GOAL(*Y BECOMES *X,
      *NO,*N,                   *CODE ,

                                VOID )

   -/
   -INTEVAL(*Y BECOMES *X,70006,*J,*K,0,*RESULT,*CODE).

+GOAL(*X=*Y,
      *NO,*N,                   MOVE(B,*X1);
                                MOVE(B1,*Y1);
                                JSP(C,EPEQ) ,

                                *DATA )

   -/
   -RTERM(*X,OUTER,*NO,*N1, WD(*X1) , *DATA1 )
   -RTERM(*Y,OUTER,*N1,*N,  WD(*Y1) , *DATA2 )
   -SHORTEN( *DATA1;*DATA2 , *DATA ).

+GOAL(FAIL,
      *N,*N,                    JRST(EPFAIL) ,

                                VOID )-/.

+GOAL(TRUE,
      *NO,*NO,                  VOID ,

                                VOID )-/.

+GOAL(VAR(*X),
      *NO,*NO,                  *CODE ,

                                VOID )
```

Comp-9

```
   -/-CHECKVAR(*X, *CODE ).

+GOAL(INCHAR(*X),
      *NO,*N,                    INCHWL(VAL);
                                 *RESULT ,

                                 VOID )

   -/
   -INTRESULT(*X,*NO,*N, *RESULT ).

+GOAL(INNBCHAR(*X),
      *NO,*N,              *L:;
                                 INCHWL(VAL);
                                 CAIN(VAL,32);
                                 JRST(*L);
                                 *RESULT ,

                                 VOID )

   -/
   -INTRESULT(*X,*NO,*N, *RESULT ).

+GOAL(SKIPCHAR(*X),
      *NO,*NO,                   *ARG;
                           *L:;
                                 INCHWL(AC1);
                                 CAIE(AC1,*X1);
                                 JRST(*L) ,


                                 VOID )

   -/
   -INTARG(*X,*X1, *ARG ).

+GOAL(OUTCHAR(*X),
      *NO,*NO,                   *ARG;
                                 OUTCHR(*X1) ,

                                 VOID )

   -/
   -INTARG(*X,*X0, *ARG )
   -INTADDR(*X0,*X1 ).

+GOAL(NEWLINE,
      *NO,*NO,                   OUTSTR(CRLF) ,

                                 VOID )-/.

+GOAL(*Q,
      *NO,*NO,                   *CODE ,

                                 VOID )

   -INTREL(*Q)-/
   -INTEVAL(*Q,70006,*J,*K,0,*RESULT, *CODE ).

+GOAL(*Q,
      *NO,*N,                    JSP(A,*PREDICATE);
                                 *ARGS ,
```

```
                                *DATA )

   -UNIV(*Q,*F.*A)-LENGTH(*A,*I)
   -FLAGPREDICATE(*F,*I,*PREDICATE)
   -RTERMS(*A,OUTER,*NO,*N, *ARGS , *DATA ).


** EVALUATION OF ALTERNATIVES IN THE BODY OF THE CLAUSE.


+DISJUNCT(*Q1 OR *Q2,
          *NO,*N,*EXIT,        MOVEI(FL,*L);
                               HRLM(FL,V(0));
                               *GOAL;
                               JRST(*EXIT);
                            *L:;
                               *DISJUNCT ,

                               *DATA )

   -/
   -GOALS(*Q1,*NO,*N1, *GOAL , *DATA1 )
   -DISJUNCT(*Q2,*N1,*N,*EXIT, *DISJUNCT , *DATA2 )
   -SHORTEN( *DATA1;*DATA2 , *DATA ).

+DISJUNCT(*Q,
          *NO,*N,*EXIT,        HRRZS(V(0));
                               *GOAL ,

                               *DATA )

   -GOALS(*Q,*NO,*N, *GOAL , *DATA ).


** CODE TO CHECK FOR A VARIABLE (EVAL PRED 'VAR').



+CHECKVAR(QQQ(*X,*EITHER),     VOID )

   -VAR(*X)-/-WARNING("OBVIOUSLY A VAR").

+CHECKVAR(QQQ(*N,MULTIPLE),    SKIPE(R1,X(*N));
                               JSP(C,CHKVAR) )-/.

+CHECKVAR(*X,                  JRST(EPFAIL) )

   -WARNING("OBVIOUSLY NOT A VAR").


** CODE FOR THE ARGUMENTS OF EVALUABLE PREDICATES.



+INTARG(*X,*X1,*CODE)
```

```
   -INTEVAL(*X,70006,*J,*K,0,*RESULT,*CODE)
   -DO( *RESULT=LIT(*N) & *X1=*N ELSE *X1=VAL(0) ).

+INTLOAD(*X,*CODE)
   -INTEVAL(*X,70006,*J,*K,0,*RESULT,*CODE1)
   -DO( *RESULT=LIT(*N) & *CODE=MOVEI(VAL,*N)
        ELSE *CODE=*CODE1 ).



+INTADDR(VAL(0),VAL)-/.

+INTADDR(*X,*ATOM)
   -UNIV(*X,*F.NIL)
   -FLAGATOM(*X,*F,*Z,*ATOM).



+INTRESULT(QQQ(*I,SINGLE),
           *N,*N,              VOID )-/.

+INTRESULT(QQQ(*NO,MULTIPLE),
           *NO,*N1,            MOVEI(R2,X(*NO));
                               JSP(C,INTASS) )

   -/-PLUS(*NO,1,*N1).

+INTRESULT(QQQ(*I,MULTIPLE),
           *N,*N,              MOVE(R2,X(*I));
                               JSP(C,INTRES) )-/.

+INTRESULT(*X,
           *N,*N,              CAIE(VAL,*X);
                               JRST(EPFAIL) )

   -UNIV(*X,(*L.*R).NIL)
   -CHIFFRE(*L)-/.

+INTRESULT(*X,
           *N,*N,              JRST(EPFAIL) )

   -WARNING("INTEGER OR VARIABLE EXPECTED").



** CODE FOR PERFORMING UNIFICATION WITH THE OUTERMOST LIST OF TERMS
** ON THE LEFT-HAND SIDE OF THE CLAUSE.



+LTERMSOUTER(*T.*A,
             *NO,*N,*MO,       *ARGCODE;
                               *ARGSCODE ,

                               *DATA )

   -PLUS(*MO,1,*M1)
   -LTERMOUTER(*T,*NO,*N1,*MO, *ARGCODE , *DATA1 )
   -LTERMSOUTER(*A,*N1,*N,*M1, *ARGSCODE , *DATA2 )
   -SHORTEN( *DATA1;*DATA2 , *DATA ).
```

```
+LTERMSOUTER(NIL,
             *N,*N,*M,          VOID ,

                                VOID ).



** CASE OF A SINGLE OCCURRENCE OF A VARIABLE.


+LTERMOUTER(QQQ(*I,SINGLE),
             *N,*N,*M,          VOID ,

                                VOID )-/.

** CASE OF THE FIRST OCCURRENCE OF A MULTIPLY OCCURRING VARIABLE.
** 'UVAR' IS CALLED IF THE ARGUMENT IS A SKELETON,REFERENCE OR VOID.


+LTERMOUTER(QQQ(*NO,MULTIPLE),
             *NO,*N1,*M,        MOVE(T,@A(*M));
                                TLNN(T,MSKMA);
                                JSP(C,UVAR);
                                MOVEM(T,V(*NO)) ,

                                VOID )

   -/-PLUS(*NO,1,*N1).

** CASE OF A SUBSEQUENT OCCURRENCE OF A VARIABLE.
** 'UREF' IS ALWAYS CALLED.


+LTERMOUTER(QQQ(*I,MULTIPLE),
             *N,*N,*M,          MOVE(B,@A(*M));
                                MOVE(B1,V(*I));
                                JSP(C,UREF) ,

                                VOID )-/.

** CASE OF AN ATOM.
** 'UATOM' IS CALLED IF THE ARGUMENT IS A REFERENCE OR VOID.


+LTERMOUTER(*T,
             *N,*N,*M,          MOVE(T,@A(*M));
                                TLNN(T,MSKMAS);
                                JSP(C,UATOM);
                                CAME(T,*ATOM);
                                JRST(FAIL) ,

                                VOID )

   -UNIV(*T,*F.NIL)-/
   -FLAGATOM(*T,*F,*Z,*ATOM).

** CASE OF A SKELETON (NON-ATOMIC,NON-VARIABLE INPUT TERM).
** 'USKEL' IS ALWAYS CALLED.
** IF THE ARGUMENT, WHEN FULLY DEREFERENCED, IS A REFERENCE,
** THEN '*ARGSCODE' IS SKIPPED.
```

```
+LTERMOUTER(*T,
            *NO,*N,*M,          MOVE(B,@A(*M));
                                JSP(C,USKEL);
                                WD(*SKEL);
                                *INITIALISE;
                                JUMPE(Y,*L);
                                *ARGSCODE;
                               *L: ,

                               *SKEL:;
                                  XWD(SKEL,*FUNCTOR);
                                  *ARGS;
                                  *DATA )

   -UNIV(*T,*F.*A)-LENGTH(*A,*I)
   -FLAGFUNCTOR(*F,*I,*FUNCTOR)
   -LTERMSINNER(*A,*NO,*N,1, *ARGS , *ARGSCODE , *DATA )
   -INITIALISE(*NO,*N, *INITIALISE ).


** UNIFICATION WITH AN INNER LIST OF TERMS AT DEPTH 1 ON THE LHS.



+LTERMSINNER(*T.*A,
             *NO,*N,*MO,        *ARG;
                                *ARGS ,

                                *ARGCODE;
                                *ARGSCODE ,

                                *DATA )

   -PLUS(*MO,1,*M1)
   -LTERMINNER(*T,*NO,*N1,*MO, *ARG , *ARGCODE , *DATA1 )
   -LTERMSINNER(*A,*N1,*N,*M1, *ARGS , *ARGSCODE , *DATA2 )
   -SHORTEN( *DATA1;*DATA2 , *DATA ).

+LTERMSINNER(NIL,
             *N,*N,*M,          VOID ,

                                VOID ,

                                VOID ).


** CASE OF A SINGLE OCCURRENCE OF A VARIABLE.


+LTERMINNER(QQQ(*NO,SINGLE),
             *NO,*N1,*M,        WD(Y(*NO)) ,

                                VOID ,

                                VOID )

   -/-PLUS(*NO,1,*N1).

** CASE OF THE FIRST OCCURRENCE OF A MULTIPLY OCCURRING VARIABLE.
** 'UVAR1' IS CALLED IF THE ARGUMENT IS A SKELETON OR REFERENCE.
```

```
+LTERMINNER(QQQ(*NO,MULTIPLE),
            *NO,*N1,*M,          WD(Y(*NO)) ,

                                 MOVE(T,@B(*M));
                                 TLNN(T,MSKMA);
                                 JSP(C,UVAR1);
                                 MOVEM(T,V(*NO)) ,

                                 VOID )

   -/-PLUS(*NO,1,*N1).

** CASE OF A SUBSEQUENT OCCURRENCE OF A VARIABLE.
** 'UREF1' IS ALWAYS CALLED.


+LTERMINNER(QQQ(*I,MULTIPLE),
            *N,*N,*M,            WD(Y(*I)) ,

                                 MOVE(T,@B(*M));
                                 MOVE(B1,V(*I));
                                 JSP(C,UREF1) ,

                                 VOID )-/.

** CASE OF AN ATOM.
** 'UATOM' IS CALLED IF THE ARGUMENT IS A REFERENCE.


+LTERMINNER(*T,
            *N,*N,*M,            WD(*ATOM) ,

                                 MOVE(T,@B(*M));
                                 TLNN(T,MSKMAS);
                                 JSP(C,UATOM);

                                 CAME(T,*ATOM);
                                 JRST(FAIL) ,

                                 VOID )

   -UNIV(*T,*F.NIL)-/
   -FLAGATOM(*T,*F,*Z,*ATOM).

** CASE OF A SKELETON.
** 'USKEL1' IS ALWAYS CALLED.


+LTERMINNER(*T,
            *NO,*N,*M,           WD(*SKEL) ,

                                 MOVE(T,@B(*M));
                                 JSP(C,USKEL1);
                                 WD(*SKEL) ,

                               *SKEL:;
                                 XWD(SKEL,*FUNCTOR);
                                 *ARGS;

                                 *DATA )
```

```
   -UNIV(*T,*F.*A)-LENGTH(*A,*I)
   -FLAGFUNCTOR(*F,*I,*FUNCTOR)
   -RTERMS(*A,INNER,*NO,*N, *ARGS ,*DATA ).
   -INITIALISE(*NO,*N, *INITIALISE ).


** CODE FOR COMPLETING UNIFICATION WITH THE FIRST
** TERM ON THE LEFT-HAND SIDE OF THE CLAUSE.


+LTERMFIRST(VOID,
            VOID,2,          VOID ,

                             VOID )-/.

+LTERMFIRST(QQQ(2,*EITHER),
            GENERAL,3,       VOID ,

                             VOID )-/.

+LTERMFIRST(*T,
            SPECIAL(*TYPE,*KEY,*ADDR),
            2,               VOID ,

                             VOID )

   -UNIV(*T,*F.NIL)-/
   -FLAGATOM(*T,*F,XWD(*TYPE,*KEY),*ADDR).

+LTERMFIRST(*T,
            SPECIAL(SKEL,*KEY,*L1),
            *N,              *INITIALISE;
                             JUMPE(Y,*L);
                             *ARGSCODE;
                          *L: ,

                          *L1:;
                             XWD(SKEL,*KEY);
                             *ARGS;
                             *DATA )

   -UNIV(*T,*F.*A)-LENGTH(*A,*I)
   -FLAGFUNCTOR(*F,*I,*KEY)
   -LTERMSINNER(*A,2,*N,1,*ARGS,*ARGSCODE,*DATA)
   -INITIALISE(2,*N,*INITIALISE).


** DATA REPRESENTING TERMS IN
**    (1) THE RIGHT-HAND SIDE (BODY) OF THE CLAUSE;
**    (2) THE LEFT-HAND SIDE OF THE CLAUSE AT DEPTH ) 1;
** THE SECOND ARGUMENT IS THE DEPTH WHICH IS EITHER 'OUTER'  OR
** 'INNER'.


+RTERMS(*T.*A,*D,
        *NO,*N,              *ARG;
```

```
                                *ARGS ,

                                *DATA )

   -RTERM(*T,*D,*NO,*N1, *ARG , *DATA1 )
   -RTERMS(*A,*D,*N1,*N, *ARGS , *DATA2 )
   -SHORTEN( *DATA1;*DATA2 , *DATA ).

+RTERMS(NIL,*D,
        *N,*N,                  VOID ,

                                VOID ).


** CASE OF AN OUTER VARIABLE HAVING A SINGLE OCCURRENCE.


+RTERM(QQQ(*I,SINGLE),OUTER,
       *N,*N,                   WD(AVOID) ,

                                VOID )-/.

** CASE OF THE FIRST OCCURRENCE OF A MULTIPLY OCCURRING VARIABLE
** AT THE OUTERMOST LEVEL.


+RTERM(QQQ(*NO,MULTIPLE),OUTER,
       *NO,*N1,                 WD(X(*NO)) ,

                                VOID )
   -/-PLUS(*NO,1,*N1).

** CASE OF THE FIRST OCCURRENCE OF A VARIABLE, OCCURRING AT AN
** INNER LEVEL.


+RTERM(QQQ(*NO,*EITHER),INNER,
       *NO,*N1,                 WD(Y(*NO)) ,

                                VOID )
   -/-PLUS(*NO,1,*N1).

** CASE OF A SUBSEQUENT OCCURRENCE OF A VARIABLE, AT THE OUTERMOST
** LEVEL.


+RTERM(QQQ(*I,MULTIPLE),OUTER,
       *N,*N,                   WD(X(*I)) ,

                                VOID )-/.

** CASE OF A SUBSEQUENT OCCURRENCE OF A VARIABLE, AT AN INNER LEVEL.


+RTERM(QQQ(*I,MULTIPLE),INNER,
       *N,*N,                   WD(Y(*I)) ,

                                VOID )-/.
```

```
** CASE OF AN ATOM.


+RTERM(*T,*D,
       *N,*N,                    WD(*ATOM) ,

                                 VOID )

   -UNIV(*T,*F.NIL)-/
   -FLAGATOM(*T,*F,*Z,*ATOM).

** CASE OF A SKELETON.


+RTERM(*T,*D,
       *NO,*N,                   WD(*SKEL) ,

                               *SKEL:;
                                  XWD(SKEL,*FUNCTOR);
                                  *ARGS;
                                  *DATA )

   -UNIV(*T,*F.*A)-LENGTH(*A,*I)
   -FLAGFUNCTOR(*F,*I,*FUNCTOR)
   -RTERMS(*A,INNER,*NO,*N, *ARGS , *DATA ).



** CODE TO INITIALISE A BLOCK OF VARIABLES TO "UNASSIGNED".


+INITIALISE(*NO,*N,              SETZM(V(*NO));
                                 HRLZI(R1,V(*NO));
                                 HRRI(R1,V(*N1));
                                 BLT(R1,V(*N2)) )

   -MOINS(*N,*NO,*I)
   -INF(4,*I)-/
   -PLUS(*NO,1,*N1)
   -MOINS(*N,1,*N2).


+INITIALISE(*N,*N,               VOID )-/.


+INITIALISE(*NO,*N,              SETZM(V(*NO));
                                 *INITIALISE )

   -PLUS(*NO,1,*N1)
   -INITIALISE(*N1,*N,*INITIALISE).



** MISCELLANY.


+FLAGATOM(*T,*F,*VAL,AT(*I))-ATOM(*T,*I,*VAL)-/.
+FLAGATOM(*T,*L.*R,XWD(INT,*T),AT(*I))
   -CHIFFRE(*L)-/
```

```
   -ATOM(*T1,*I0,*VAL1)-/
   -PLUS(*I0,1,*I)
   -AJOUT(+ATOM(*T,*I,XWD(INT,*T)).NIL).
+FLAGATOM(*T,*F,XWD(ATOM,*FUNCT),AT(*I))
   -ATOM(*T1,*I0,*VAL1)-/
   -PLUS(*I0,1,*I)
   -FLAGFUNCTOR(*F,0,*FUNCT)
   -AJOUT(+ATOM(*T,*I,XWD(ATOM,*FUNCT)).NIL).

+ATOM("NIL",0,XWD(ATOM,0)).

+FLAGFUNCTOR(*F,*N,*I)-FUNCTOR(*F,*N,*I)-/.
+FLAGFUNCTOR(*F,*N,*I)
   -SUPP(+FUNCTORCOUNT(*I).NIL)
   -PLUS(*I,1,*I1)
   -AJOUTC(+FUNCTOR(*F,*N,*I).NIL)
   -AJOUT(+FUNCTORCOUNT(*I1).NIL).

+FUNCTOR("NIL",0,0).
+FUNCTOR(".",2,1).

+FUNCTORCOUNT(2).

+FLAGPREDICATE(*F,*N,PRED(*I))-PREDICATE(*F,*N,*I)-/.
+FLAGPREDICATE(*F,*N,PRED(*I))
   -PREDICATE(*F1,*N1,*I0)-/
   -PLUS(*I0,1,*I)
   -AJOUT(+PREDICATE(*F,*N,*I).NIL).

+PREDICATE("GOAL",0,0).
+PREDICATE("SORTER",1,SORTE).
+PREDICATE("PUNIV",2,PPUNI).

+FLAGCLAUSE(CL(*I))
   -SUPP(+CLAUSECOUNT(*I0).NIL)-/
   -PLUS(*I0,1,*I)
   -AJOUT(+CLAUSECOUNT(*I).NIL).

+CLAUSECOUNT(0).

+MARKVARS(*T)-VAR(*T)-/-EQUAL(*T,QQQ(*I,*Z)).
+MARKVARS(QQQ(*I,MULTIPLE))-/.
+MARKVARS(*T)-UNIV(*T,*F.*A)-MARKVARSLIST(*A).

+MARKVARSLIST(*T.*A)-MARKVARS(*T)-MARKVARSLIST(*A).
+MARKVARSLIST(NIL).

+COMPLETERHS(*P WHERE *Q,*P WHERE *Q)-/.
+COMPLETERHS(*P,*P WHERE TRUE).

+CHECKSLASH(*P!,*P, HRRZS(V(0)) )-/.
+CHECKSLASH(*P,*P, HRLM(FL,V(0)) ).

+LENGTH(*T.*A,*N1)-LENGTH(*A,*N)-PLUS(*N,1,*N1).
+LENGTH(NIL,0).

+MAX(*X,*Y,*Y)-INF(*X,*Y)-/.
+MAX(*X,*Y,*X).


+MIN(*X,*Y,*Y)-INF(*Y,*X)-/.
```

```
+MIN(*X,*Y,*X).

+SHORTEN( VOID;*X ,*X)-/.
+SHORTEN( *X;VOID ,*X)-/.
+SHORTEN( *X ,*X).

+DO(*X ELSE *Y)-DO(*X)-/.
+DO(*X ELSE *Y)-/ -DO(*Y).
+DO(*X&*Y)-/-DO(*X)-DO(*Y).
+DO(*X=*Y)-/-EQUAL(*X,*Y).
+DO(*X)*Y)-/-INF(*Y,*X).
+DO(*X OR *Y)-DO(*X).
+DO(*X OR *Y)-/-DO(*Y).
+DO(*X)-*X.

+EQUAL(*X,*X).



** EXTERNAL INTERFACE.



+COMPILECLAUSES
   -ASSERTED(*C)
   -CLAUSE(*C,*F,*N,*ARG1,*CODE)
   -FLAGCLAUSE(*CL)
   -ASSEMBLE(*CODE,*CL)
   -OUTPUT( *CL:;*CODE )
   -FLAGPREDICATE(*F,*N,*PRED)
   -RECORDCLAUSE(*PRED,*CL:*ARG1)
   -FAIL.
+COMPILECLAUSES.

+COMPILE-COMPILECLAUSES-FAIL.
+COMPILE
   -BLOCK(*PRED,*BLOCK)
   -NOTCOMPILED(*PRED)
   -AJOUT(+COMPILED(*PRED).NIL)
   -COMPILEBLOCK(*BLOCK,*CODE)
   -ASSEMBLE(*CODE,*PRED)
   -OUTPUT( *PRED:;*CODE )
   -FAIL.
+COMPILE-OUTPUT( FNTAB: )-FAIL.
+COMPILE
   -FUNCTOR(*F,*N,*I)
   -OUTPUT( XWD(*N,NAME(*I)) )
   -FAIL.
+COMPILE
   -FUNCTOR(*F,*N,*I)
   -CODESTRING(*F,*STRING)
   -OUTPUT( NAME(*I):;*STRING )
   -FAIL.
+COMPILE
   -ATOM(*T,*I,*VAL)
   -OUTPUT( AT(*I):;*VAL )
   -FAIL.
+COMPILE-SORCHA("   END START")-LIGNE.

+NOTCOMPILED(*P)-COMPILED(*P)-/-FAIL.
+NOTCOMPILED(*P).
```

```
+DELETECLAUSES-ASSERTED(*C)-SUPP(+ASSERTED(*C).NIL)-FAIL.
+DELETECLAUSES.

** IDEALLY 'ASSEMBLE' WOULD GENERATE INPUT FOR THE LOADER DIRECTLY.
** FOR THE TIME BEING, THE JOB IS DELEGATED TO ~~BCPL~~. MACRO.
** ~~THE OBSCURE CODING IS TO CONSERVE STORAGE ON PROLOG'S STACK.~~

+ASSEMBLE(*CODE,*L)-ASSEMBLE(*CODE,*L,0,*N)-/.

+ASSEMBLE(*X;*Y,*L,*NO,*N)-/
   -ASSEMBLE(*X,*L,*NO,*N1)
   -ASSEMBLE(*Y,*L,*N1,*N).
+ASSEMBLE(LAB(*L,*N):,*L,*N,*N)-/.
+ASSEMBLE(*LAB:,*L,*N,*N)-/.
+ASSEMBLE(VOID,*L,*N,*N)-/.
+ASSEMBLE(*X,*L,*NO,*N)
   -PLUS(*NO,1,*N).

+OUTPUT(*X)-OUTPUT1(*X)-FAIL.
+OUTPUT(*X).

+OUTPUT1(*X;*Y)-/-OUTPUT(*X)-OUTPUT(*Y).
+OUTPUT1(VOID)-/.
+OUTPUT1(LAB(*L,*N):)-/.
+OUTPUT1(*L:)-/
   -PUTITEM(*L)
   -SORCHA(":").
+OUTPUT1(*X)-UNIV(*X,*F.*A)
   -SORCHA("  ")
   -SORCHA(*F)
   -SORCHA(" ")
   -PUTITEMS(*A)
   -LIGNE.

+PUTITEMS(NIL)-/.
+PUTITEMS(*X.NIL)-/-PUTITEM(*X).
+PUTITEMS(*X.*Y.NIL)-/-PUTITEM(*X)-SORCHA(",")-PUTITEM(*Y).

+PUTITEM(*X.*Y)-/-SORCHA("""")-SORCHA(*X.*Y)-SORCHA("""").
+PUTITEM(@*X)-/-SORCHA("@")-PUTITEM(*X).
+PUTITEM(LAB(*L,*N))-/-PUTITEM(*L)-SORCHA("+")-SORTER(*N).
+PUTITEM(PRED(*N))-/-SORCHA("P")-SORTER(*N).
+PUTITEM(DBL(*N))-/-SORCHA("DBL")-SORTER(*N).
+PUTITEM(AT(*N))-/-SORCHA("A")-SORTER(*N).
+PUTITEM(CL(*N))-/-SORCHA("C")-SORTER(*N).
+PUTITEM(NAME(*N))-/-SORCHA("N")-SORTER(*N).
+PUTITEM(AC(*I))-/-RESTE(*I,7,*I0)-SORCHA("AC")-SORTER(*I0).
+PUTITEM(*X:*Y)-/
   -PUTITEM(*X)
   -SORCHA("(")
   -PUTITEM(*Y)
   -SORCHA(")").
+PUTITEM(-*X)-/-SORCHA("-")-PUTITEM(*X).
+PUTITEM(*X)-UNIV(*X,*F.NIL)-SORCHA(*F).
+PUTITEM(*X)-UNIV(*X,*F.*Y.NIL)
   -PUTITEM(*Y)-SORCHA("(")-SORCHA(*F)-SORCHA(")").

+WARNING(*X)-SORCHA(";WARNING:")-SORCHA(*X)-LIGNE.


+ERROR(*X)-SORCHA(";ERROR:  ")-SORCHA(*X)-LIGNE.
```

```
+RECORDCLAUSE(*PRED,*CL)
   -BLOCK(*PRED,*LIST)-/
   -AJOUT(+BLOCK(*PRED, *LIST;*CL ).NIL).
+RECORDCLAUSE(*PRED,*CL)
   -AJOUT(+BLOCK(*PRED, VOID;*CL ).NIL).

** GENERATION OF CODE TO MAP A GOAL ONTO A SET OF POTENTIALLY
** MATCHING CLAUSES.

+COMPILEBLOCK(*LIST;*CL:VOID, JSP(C,INTROO);
                              *CODE;
                              JSP(FL,*CL);
                              WD(0) )

   -/-SIMPLEBLOCK(*LIST,*CODE).

+COMPILEBLOCK(*LIST;*X,       JSP(C,INTRO);
                              *SECTIONS;
                              *SECTION;
                              WD(0) )

   -DO( *X=(*CL:GENERAL) & GENSECTION(*LIST;*X,*LISTO,*SECTION) &
                           SPSECTIONS(*LISTO,*SECTIONS)
   ELSE              FINALSPSECTION(*LIST;*X,*LISTO,*SECTION) &
                     GENSECTIONS(*LISTO,*SECTIONS) ).



+SIMPLEBLOCK(*LIST;*CL:VOID,  *CODE;
                              JSP(FL,*CL) )

   -SIMPLEBLOCK(*LIST,*CODE).

+SIMPLEBLOCK(VOID,            VOID ).


+GENSECTIONS(VOID,            VOID )-/.

+GENSECTIONS(*LIST,           *SECTIONS;
                              *SECTION )

   -GENSECTION(*LIST,*LISTO,*SECTION)
   -SPSECTIONS(*LISTO,*SECTIONS).


+SPSECTIONS(VOID,             VOID )-/.

+SPSECTIONS(*LIST,            *SECTIONS;
                              *SECTION )

   -SPSECTION(*LIST,*LISTO,*SECTION)
   -GENSECTIONS(*LISTO,*SECTIONS).


+GENSECTION(*LIST;*CL:GENERAL,
            *LISTO,           *CODE;
                              JSP(FL,*CL) )
```

```
   -/-GENSECTION(*LIST,*LIST0,*CODE).

+GENSECTION(*LIST,*LIST,      VOID ).



+SPSECTION(*LIST,
           *LIST0,            JSP(C,*SSECRTN);
                              WD(*L3);
                         *L1:;JSP(C,STARG1);
                         *L2:;JRST(*L4);
                              *NONREFCODE;
                         *L3:;*REFCODE;
                              HRRZ(R1,V(-1));
                              MOVEM(R1,V(2));
                         *L4:   )

   -SPSECTIONA(*LIST,*LIST0,*N,*SUBLIST,*REFCODE)
   -SPSECTIONB(*N,*SUBLIST,*L1,*L2,*NONREFCODE)
   -SPSECTIONRTN(*LIST0,*N,*SSECRTN).



+FINALSPSECTION(*LIST,
                *LIST0,       JSP(C,*SSECRTN);
                              WD(*L3);
                         *L1:;WD(0);
                         *L2:;JRST(FAILB);
                              *NONREFCODE;
                         *L3:;*REFCODE )

   -SPSECTIONA(*LIST,*LIST0,*N,*SUBLIST,*REFCODE)
   -SPSECTIONB(*N,*SUBLIST,*L1,*L2,*NONREFCODE)
   -SPSECTIONRTN(*LIST0,*N,*SSECRTN).



+SPSECTIONRTN(VOID,*N,SSECL0)-INF(*N,5)-/.
+SPSECTIONRTN(VOID,*N,SSECT0)-/.
+SPSECTIONRTN(*LIST,*N,SSECL)-INF(*N,5)-/.
+SPSECTIONRTN(*LIST,*N,SSECT)-/.



+SPSECTIONA(*LIST;*CLAUSE,
            *LIST0,*N,
            *SUBLIST;*CLAUSE, *CODE;
                              *CALL )

   -EQUAL(*CLAUSE,*CL:SPECIAL(*TYPE,*KEY,*ARG1))-/
   -DO( *TYPE=SKEL & *CALL= ( MOVEI(R1,*ARG1);
                              JSP(C,ASSSK1);
                              JSP(FL,*CL) )

   ELSE               *CALL= ( MOVE(R1,*ARG1);
                              MOVE(R2,V(-1));
                              MOVEM(R1,R2(0));
                              JSP(FL,*CL) ) )

   -SPSECTIONA(*LIST,*LIST0,*N1,*SUBLIST,*CODE)
   -PLUS(*N1,1,*N).
```

```
+SPSECTIONA(*LIST,*LIST,0,VOID,VOID).


+SPSECTIONB(*N,*LIST,
            *L1,*L2,          *TESTS;
                              JRST(@*L2);
                              *PROCS )

   -INF(*N,5)-/
   -GROUPLIKECLAUSES(*LIST,*LIST1)
   -SEQTESTS(*LIST1,*L1,*TESTS,*PROCS).

+SPSECTIONB(*N,*LIST,
            *L1,*L2,          WD(*MASK);
                              WD(@(*L:R1));
                           *L:;
                              *TABLE;
                              *CODE )

   -MOINS(*N,1,*N1)
   -CHOOSEMASK(*N1,*BIT.*BITS)
   -MULT(*BIT,2,*BIT1)-MOINS(*BIT1,1,*MASK)
   -ARRAYTESTS(*BIT.*BITS,*LIST,*L1&*L2,*TABLE,*CODE).


** A 'BIT' IS AN INTEGER OF THE FORM 2**N.


+ARRAYTESTS(*BIT.*BITS,*LIST,
            *LABELPAIR,       *TABLE1;
                              *TABLE2 ,

                              *CODE1;
                              *CODE2 )

   -SPLITONBIT(*BIT,*LIST,*LIST1,*LIST2)
   -ARRAYTESTS(*BITS,*LIST1,*LABELPAIR,*TABLE1,*CODE1)
   -ARRAYTESTS(*BITS,*LIST2,*LABELPAIR,*TABLE2,*CODE2).

+ARRAYTESTS(NIL,VOID,
            *L1&*L2,          WD(@*L2) ,

                              VOID )-/.

+ARRAYTESTS(NIL,*LIST,
            *L1&*L2,          WD(*L) ,

                           *L:;
                              *TESTS;
                              JRST(@*L2);
                               *PROCS )

   -GROUPLIKECLAUSES(*LIST,*LIST1)
   -SEQTESTS(*LIST1,*L1,*TESTS,*PROCS).
```

```
+SPLITONBIT(*BIT,VOID,VOID,VOID)-/.
+SPLITONBIT(*BIT,*LIST;*CLAUSE,*LIST1,*LIST2)
   -SPLITONBIT(*BIT,*LIST,*LIST1A,*LIST2A)
   -EQUAL(*CLAUSE,*CL:SPECIAL(*TYPE,*KEY,*ARG1))
   -SELECTBIT(*BIT,*KEY,*X)
   -DO( *X=0 & *LIST1=(*LIST1A;*CLAUSE) & *LIST2=*LIST2A
     OR *X=1 & *LIST1=*LIST1A & *LIST2=(*LIST2A;*CLAUSE) ).

+SELECTBIT(*BIT,*KEY,*X)
   -MULT(*BIT,2,*BIT1)
   -RESTE(*KEY,*BIT1,*REM)
   -DIV(*REM,*BIT,*X).

+CHOOSEMASK(1,1.NIL)-/.
+CHOOSEMASK(*N,*BIT1.*BIT2.*BITS)
   -DIV(*N,2,*N1)
   -CHOOSEMASK(*N1,*BIT2.*BITS)
   -MULT(*BIT2,2,*BIT1).

+GROUPLIKECLAUSES(*LIST0;*CLAUSE,*LIST)
   -GROUPLIKECLAUSES(*LIST0,*LIST1)
   -DO( INSERTCLAUSE(*LIST1,*CLAUSE,*LIST)
   ELSE *LIST=(*LIST1;*CLAUSE) ).

+GROUPLIKECLAUSES(VOID,VOID).

+INSERTCLAUSE(*LIST;*CLS:SPECIAL(*TYPE,*KEY,*ARG1),
                     *CL:SPECIAL(*TYPE,*KEY,*ARG1A),
               *LIST;(*CLS;*CL):SPECIAL(*TYPE,*KEY,*ARG1))-/.
+INSERTCLAUSE(*LIST0;*CLAUSE0,*CLAUSE,*LIST;*CLAUSE0)
   -INSERTCLAUSE(*LIST0,*CLAUSE,*LIST).



+SEQTESTS(*LIST;*CLS:SPECIAL(*TYPE,*KEY,*ARG1),
          *L1,
                                *TESTS;
                                CAMN(R2,*ARG1);
                                JRST(*L) ,

                                *PROCS;
                                *PROC )

   -SEQTESTS(*LIST,*L1,*TESTS,*PROCS)
   -LIKECLAUSES(*CLS,*TYPE,*L1,*L,*PROC).

+SEQTESTS(VOID,*L1,VOID,VOID).


+LIKECLAUSES(*CLS;*CL,
             *TYPE,*L1,*L, *L:;
                                *PROC;
                                MOVEI(FL,*L1);
                                JRST(*CL) )


   -/
   -LIKECLAUSES1(*CLS,*TYPE,*PROC).

+LIKECLAUSES(*CL,
             *TYPE,*L1,*CL,   VOID ).
```

```
+LIKECLAUSES1(*CLS;*CL,
              *TYPE,          *PROC;
                              JSP(FL,*CL);
                              *RELOAD )

   -/
   -LIKECLAUSES1(*CLS,*TYPE,*PROC)
   -RELOADIFSKEL(*TYPE,*RELOAD).

+LIKECLAUSES1(*CL,
              *TYPE,          JSP(FL,*CL);
                              *RELOAD )

   -RELOADIFSKEL(*TYPE,*RELOAD).

+RELOADIFSKEL(SKEL, JSP(C,RELDSK) )-/.
+RELOADIFSKEL(*TYPE, VOID ).

+CODESTRING(*F, ASCIZ(*F) ).

-LET(*X;*Y)-/+LET(*X)+LET(*Y).
-LET(*X)-AJOUTC(+ASSERTED(*X).NIL).

-TRACE-SUPP(+NOTRACE.NIL).
-NOTRACE-AJOUT(+NOTRACE.NIL).

+FIN.
-TTY.
```

.