MARK
DIVECCHIO

```
#########################################CMU#########################

G  OPER.NB03 26 OCT 68  16:15:53  AND     PAGES:  30  TIME:  3
1157   20352006     00003011403          MD05


ASSEMBLY DATE     17 NOV    FIX DATE        18 MAR 00:42:31

   AN FILE 2/P;GETTOS;PRINTS;RESET1;


Y013 MD05       2 /  1       1774    12083     117                  17075

        CO COMPLETE CHESS PROGRAM--BY JOSEPH S. RUBENFELD
        CO                        --AND ROBERT W WALKER
        CO VERSION 5.
        CO COPYRIGHT 1967 BY JOSEPH S RUBENFELD
        CO UNLIMITED LOOK AHEAD.
           BEGIN
        SY SEGMENT 5,2
           LABEL RDER,ERR,GUN;
           INTEGER F,I,J,K,L,N,R,S,T,
                   FRO,SAVE,RPT,KL,
                   DEBUG, PICK, DIR,
                   MT,SIDE,PASS,
                   LEV,MAN,P1,POINT,
                   T1,T2,T3,T4,TOSQ,
                   TTY, TEMP1, TEMP2, TEMP3, TEMP4,
                   ATAK,INIT,H0,L1,VAL;
           LOGIC WORB;
           BOOLEAN ARRAY CASTLE[-1:4];
           BOOLEAN PRINT.OR.NOT;
           BOOLEAN CARD.MOVE ;
           HALF ARRAY CC[0:6];
           LOGIC ARRAY G[-6:6],G1[0:1],PNAME[0:25];
           HALF REASON;
           HALF ARRAY V[0:7];
           SWITCH WHERE←REED.A.MOVE,MAKE.A.MOVE;
           INTEGER HERE;
           INTEGER ARRAY B[-12:530],
           TABLE[1:2,1:4,1:1000],
           QP[0:9 ],P[1:2],
           DIRE[1:5,0:50],
           DC, ! NAME OF LIST OF DISCOVERED CHECKS
           CHECK,                    !NUMBER OF TIMES I'M IN CHECK.
           KSQR,QSQR[1:2],
           NAM,ROW,ALPHA[0:30],  ! FOR PICK A AMOVE 5.
           LIST[-100:1000];
           HALF ARRAY C,X[1:2,0:25];  ! COEFFICIENTS,FACTORS


           PROCEDURE DIRECTORY;
           BEGIN INTEGER I,J;
           PRINT(<E,'DIRECTORY OF TABLES',E>);
           FOR I←0 STEP 1 WHILE DIRE[1,I]>0 DO
```

```
BEGIN
NAME(J→5(DIRE[J,I]));
PRINT(→5<-6D,3B>,<E>);
END;
NAME(POINT); PRINT(<'POINT = ',-3D.,2E>);
END OF PRINT DIRECTORY ROUTINE;
!!
PROCEDURE J151(NAM); INTEGER NAM;
BEGIN
INTEGER I;
NAME( NAM);
PRINT(<'LIST', -3D,' FOLLOWS:',E>);
NAME($FOR I←LIST[1-NAM] STEP 1 UNTIL
LIST[-NAM] DO$
(LIST[I]));
PRINT(→$ (LIST[-NAM]-LIST[1-NAM]+1)$
     <-5D,4B>,<2E>);
END OF PRINT LIST NAM ROUTINE;

PROCEDURE J152;
BEGIN
NAME($FOR I←1,-1,3,2,0,4 DO$(CASTLE[I]));
PRINT(<10C,'KING',20C,'LEFT',30C,'RIGHT',E,
'WHITE',10C>,→3<5T,5B>,<E,'BLACK'>,
<10C>,
→3<5T,5B>,<2E>);
END OF PRINT CASTLE INFORMATION;

PROCEDURE J153(NAM,K);
VALUE NAM,K;
INTEGER NAM,K;
BEGIN
INTEGER I,J,ROW;
NAME(NAM,K);
PRINT(<'LIST',-3D,' FOR SIDE ',2D,' FOLLOWS:',2E>);
PRINT(<'TO',3B,'FRO',4B,'MAN',
       4B,'VALUE',2E>);
FOR I←LIST[1-NAM] STEP 1 UNTIL LIST[-NAM] DO
BEGIN
ROW←LIST[I];
NAME(J→2(TABLE[K,J,ROW]),
      G[TABLE[K,3,ROW]],
        TABLE[K,4,ROW]);
PRINT(<2D,6D,4B,2A,-8D,E>);
END;
END OF PRINT A LIST OF MOVES;

PROCEDURE J154;   ! PRINT FACOTRS
BEGIN   !
INTEGER I,J,K;
NAME(G[MAN],TOSQ,I→$(X[1,0]+1)$(X[1,I]→C[1,I]),
HO,I→$(X[2,0]+1)$(X[2,I]→C[2,I]));
PRINT(<2A,'→',2D.6D>,
→$(X[1,0]+1)$<-3D.3D,1->,<E>,
<-4D,8C>,→$(X[2,0]+1)$<-3D.2D,1->,<E>);
```

```
END OF J154;
!!
PROCEDURE ERROR;
BEGIN
INTEGER K;
PRINT(<P,'TABLE OF MOVES.', 2E,15C,'WHITE',2E>);
FOR K←1,2 DO
BEGIN
NAME(J→$P[K]$
    (J,TABLE[K,1,J],
TABLE[K,2,J], G[TABLE[K,3,J]],
TABLE[K,4,J]));
PRINT(→$P[K]$
<3D.,3B,2D,5B,2D,5B,2A,4B,-3D,E>);
PRINT(<P>);
IF K=1 THEN
PRINT(<15C,'BLACK',2E>);
END;
END OF ERROR DUMP PROCEDURE;
!!
PROCEDURE BOARDWRITE;
BEGIN
INTEGER I,J,L,L1;
IF TTY≠1 THEN PRINT(<15C>) ELSE PRINT(<2C>);
PRINT(→41<'-'>,<E>);
FOR L1←1 STEP 2 UNTIL 8 DO BEGIN
NAME($FOR I←L1 STEP 1 UNTIL L1+1 DO$
(J→8( IF B[10*I+J]=0 THEN G[B[10*I+J]]
ELSE G1[(I+J) MOD 2])));
FOR L←2,3,2,1,4,5,4,1 DO BEGIN
IF TTY≠1 THEN PRINT(<15C>) ELSE PRINT(<2C>);
BEGIN
SWITCH S← V,W,X,Y,Z;  GO TO S[L];
V:   PRINT(→$(IF I<8 THEN 1 ELSE 0)$
        (<'-----'>,→7<'+----'>,<I-'>),
     →$(IF I=8 THEN 1 ELSE 0)$
        (→41<'-'>), <E>);
     GO TO THE;
W:    PRINT(<'|'>, →4<4B, '|////|'>,<E>) ; GO TO THE;
X:    PRINT(<'|'>, →4<B, 2A,B,'|/', 2A,'/|'>,<E>); GO TO THE;
Y:    PRINT(→4<'|////|', 4B>,<'|',E>); GO TO THE;
Z:    PRINT(<'|'>,→4<'/', 2A,'/|',B, 2A,B,'|'>,<E>); GO TO THE;
THE:  END;END;
END;
PRINT(<E>);
END OF BOARDWRITE;

PROCEDURE SUMMARIZE;
BEGIN
INTEGER X,D,R1,R2,F1,F2,T0,I,R;
INTEGER T1,T2,H,J;
LOGIC ARRAY FILES [1:8];
AL   PROCEDURE LINE;PRINT(→54<'-'>,<E>);
FILES[1]←'KR'; FILES[2]←'KN'; FILES [3]←'KB';
FILES[4]←' K'; FILES[5]←' Q'; FILES[6]←'QB';
```

add 7 to G
add 1 to G1
add 13 to B

```
   FILES[7]←'QN'; FILES[8]←'QR';
      PRINT(<P>);
AL PRINT(1);
      LINE;
      PRINT(<'| MOVE |',13C,'W H I T E',30C,'|',37C,
AL 'B L A C K',54C,'|                    ',E>);
AL PRINT(<'|NUMBER|'>,→21<'-'>,<'|'>,→23<'-'>,
AL <'|'.E>);
AL PRINT(<'|',8C,'| PROGRAM |  REGULAR  | PROGRAM  |',
AL '  REGULAR  |'          ,E>);
   LINE;
   X←0;  T0←B[120];
   FOR D←121 STEP 2 UNTIL T0 DO
   BEGIN
   X←X+1;
   H←B[D]; J←B[D+1];
   T1←↓(H*.01) MOD 100;  T2←↓(J*.01) MOD 100;
   F1←T1 MOD 10;  F2← - (T2 MOD 10);
   R1←.1*T1;  R2:=9+↓(.1*T2);
   IF D=T0 THEN J←R2←F2←T2←0;
   NAME(X,G[↓(H*.0001)],T1,G[↓(H*.0001)],FILES[F1],R1,
        G[↓(J*.0001)],T2,G[↓(J*.0001)],FILES[F2],R2);
AL PRINT(<'|',4D,8C,'|',10C,3A,'→',3D,18C,'|',22C,1A,
AL '-',2A,1D,30C,'|',2R,3A,'→',3D,42C,'|',3R.1A,'-',
AL 2A,1D,54C,'|',                  E>);
   LINE;
   END;
   END PROCEDURE SUMMARIZE;

   PROCEDURE LYST(NAM,ITEM);
   VALUE ITEM; INTEGER NAM,ITEM;
   BEGIN
   INTEGER I,N;
   OWN INTEGER POINT;
   IF NAM<0 THEN
CO ERASE LIST NAM
   BEGIN
   IF LIST[NAM]>1000 THEN
   PRINT(<'***LIST OVERFLOW***',2E>);
   LIST[NAM]←LIST[1+NAM]←0;
   N←NAM;          !CAUSE NAM IS CALL BY NAME
   RESET:
   IF -N=POINT ∧ LIST[N]=0 THEN
   BEGIN
   POINT←-( N←N+2);
   IF POINT ¬<1 THEN
   GO TO RESET;
   END;
   GO TO THE END;
   IF NAM =0 THEN
CO CREATE A NEW LIST NAMED NAM.
   BEGIN
   IF POINT<100 THEN
     BEGIN
     NAM←(POINT←POINT+2);
```

```
    OK:
      LIST[1-NAM]← 1+ (LIST[-NAM]←LIST[2-NAM]);
      END ELSE
      BEGIN
    FOR POINT←2 STEP 2 WHILE LIST[-POINT]←=0 DO;
    IF POINT<100 THEN
    BEGIN
    NAM←POINT;
    POINT←100;
    GO TO OK;
    END ELSE
    BEGIN
    PRINT(<'*** LIST NAME OVERFLOW ***',2E>);
    GO TO THE;
          END;
        END;
      END;
CO  ADD ITEM TO LIST NAM.
    IF NAM = POINT THEN
    GO TO ADD;
    FOR N←-NAM-2 STEP -2
    WHILE LIST[N]=0 DO;
    IF LIST[1+N]>LIST[-NAM]+1 THEN !THERE IS ROOM
    GO TO ADD
    ELSE
    BEGIN
    IF POINT -<100 THEN
    BEGIN
    PRINT(<E,'** LISTS OVERLAP  **',2E>);
    GO TO ADD;
    END;
    N← LIST[-POINT]-LIST[1-NAM]+1;
    POINT←POINT+2;
    FOR I←LIST[1-NAM] STEP 1 UNTIL LIST[-NAM] DO
    LIST[I+N]←LIST[I];
    LIST[-NAM]←LIST[1-NAM]←0;
    NAM←POINT;
    LIST[1-NAM]←1+LIST[2-NAM];
    LIST[-NAM]←I+N-1;
    END;
    ADD:
    LIST[(LIST[-NAM]←LIST[-NAM]+1)]←ITEM;
    THE:
    END OF LYST PROCESSING ROUTINE;

    PROCEDURE TEST(RELATION,OPERAND,COLUMN,K);
    VALUE RELATION,OPERAND,COLUMN,K;
    INTEGER RELATION, OPERAND,COLUMN,K;
    BEGIN
    INTEGER I;
    SWITCH REL←R1,R2,R3,R4,R5;
    H0←0;
    GO TO REL[RELATION];
    R1:
    FOR I←1 STEP 1 UNTIL P[K] DO
```

```
    IF TABLE[K,COLUMN,I]¬>0 THEN LYST(H0,I);
    GO TO EXIT;
    R2:
    FOR I←1 STEP 1 UNTIL P[K] DO
    IF TABLE[K,COLUMN,I]¬<0 THEN LYST(H0,I);
    GO TO EXIT;
    R3:
    FOR I←1 STEP 1 UNTIL P[K] DO
    IF TABLE[K,COLUMN,I]¬<OPERAND THEN LYST(H0,I);
    GO TO EXIT;
    R4:
    FOR I←1 STEP 1 UNTIL P[K] DO
    IF TABLE [K,COLUMN,I]¬>OPERAND THEN LYST(H0,I);
    GO TO EXIT;
    R5:
    FOR I←1 STEP 1 UNTIL P[K] DO
    IF TABLE[K,COLUMN,I]=OPERAND THEN LYST(H0,I);
    EXIT:
AL END OF TEST;

  PROCEDURE SEARCH( INITIAL,FINAL,ITEM,I,K)
            VALUE INITIAL, FINAL, ITEM, I,K;
            INTEGER INITIAL,FINAL,ITEM,I,K;
    BEGIN
    FOR H0←INITIAL STEP 1 UNTIL FINAL DO
    IF TABLE[K,I,H0]=ITEM THEN GO TO EXIT;
    H0←0;
    EXIT:
AL END OF PROCEDURE SEARCH;

  PROCEDURE COUNT(RELATION,OPERAND,COLUMN,K);
    VALUE RELATION,OPERAND,COLUMN,K;
    INTEGER RELATION, OPERAND,COLUMN,K;
    BEGIN
    INTEGER I;
    SWITCH REL←R1,R2,R3,R4,R5;
    H0←0;
    GO TO REL[RELATION];
    R1:
    FOR I←1 STEP 1 UNTIL P[K] DO
    IF TABLE[K,COLUMN,I]¬>0 THEN H0←H0+1;
    GO TO EXIT;
    R2:
    FOR I←1 STEP 1 UNTIL P[K] DO
    IF TABLE [K,COLUMN,I]¬<0 THEN H0←H0+1;
    GO TO EXIT;
    R3:
    FOR I←1 STEP 1 UNTIL P[K] DO
    IF TABLE[K,COLUMN,I]¬<OPERAND THEN H0←H0+1;
    GO TO EXIT;
    R4:
    FOR I←1 STEP 1 UNTIL P[K] DO
    IF TABLE[K,COLUMN,I]¬>OPERAND THEN H0←H0+1;
    GO TO EXIT;
    R5:
```

```
   FOR I←1 STEP 1 UNTIL P[K] DO
   IF TABLE[K,COLUMN,I]=OPERAND THEN H0←H0+1;
   EXIT:
AL END OF COUNT ROUTINE;

   PROCEDURE ENTER(Y1,Y2,Y3,Y4); VALUE Y1,Y2,Y3,Y4;
   INTEGER Y1,Y2,Y3,Y4;
   BEGIN
   INTEGER  PK,T2;
   PK←P[K]←P[K]+1;
   TABLE[K,1,PK]←Y1;
   TABLE[K,2,PK]←Y2;
   TABLE[K,3,PK]←Y3;
   TABLE[K,4,PK]←SIGN(Y4)*V[ABS(Y4)];
   END PROCEDURE ENTER;
   !!
   INTEGER PROCEDURE REPEATS;
   BEGIN ! FUNCTION TO TELL HOW MANY TIMES THE CURRENT BOARD POSITION
CO HAS OCURRED PREVIOUSLY.  ASSUMES THE FOLLOWING:
CO 1. B[121]...B[500] CONTAINS ALL THE MOVES IN THE FORM MTTFF*SIGN(M)
CO     WHERE M=MAN TYPE, TT=TO SQUARE, FF=FROM SQUARE.
CO 2. B[120] CONTAINS THE B WITH THE LATEST MOVE.
CO 3. B[-11] CONTAINS THE B WITH THE LAST IRREVERSIBLE MOVE.
   INTEGER ARRAY MOVE[1:2,-1:50];
   COMMENT: 50 REVERSIBLE MOVES = STALEMATE;
   INTEGER I,J,K,L;
   REPEATS←0;
   IF B[120]-B[-11]>7 THEN
   BEGIN
AL     J←0;
       FOR I←B[-11]+1 STEP 1 UNTIL B[120] DO
       BEGIN
          J←J+1;
          MOVE[1,J]←B[I]*.01; ! TO SQ
AL        MOVE[2,J]←100*↓(B[I]*.0001)←B[I] MOD 100; !FRO
       END;
       L←K←J;
       FOR J←J STEP -1 UNTIL 1 DO
       BEGIN
          MOVE[1,-1]←MOVE[1,0]←MOVE[2,J];
   I←K-((K-J) MOD 2)+2;
   FOR I←I-2 WHILE MOVE[1,I]≠MOVE[2,J] DO;  ! NOTHING
          IF I<1 THEN GO TO RETURN; ! NO MATCH
          MOVE[1,I]←0;
          L←MIN(L,I);
          IF J=L THEN
          BEGIN
             REPEATS←REPEATS+1;
             K←J-1
          END
       END
   END;
   RETURN:
   END OF PROCEDURE REPEATS;
```

```
PROCEDURE RECORD.MOVE;
BEGIN
B[(B[120]←B[120]+1)]←SIGN(MAN)*(FRO←100*(TOSQ←100*ABS(MAN)));
END OF RECORD MOVE;

PROCEDURE PAWN;
BEGIN
INTEGER INC,LIM1,LIM2,R,F;
R←FRO/10; F←FRO-10*R;
IF K=1 THEN BEGIN INC←1;
                  LIM1←8; LIM2←2; END
     ELSE BEGIN INC←-1;
                  LIM1←1; LIM2←7; END;
IF R≠LIM1 THEN
BEGIN IF B[10*(R←INC)+F] =0 THEN
  BEGIN IF R=LIM2 THEN
    BEGIN IF B[10*(R←2*INC)+F]=0 THEN
ENTER(10*(R←2*INC)+F,FRO,MT,0);
  END;
ENTER(10*(R←INC)+F,FRO,MT,0);
END;
FOR LIM1←8,1 DO
BEGIN
LIM2← IF LIM1=8 THEN 1 ELSE -1;
       IF F ≠LIM1 THEN
ENTER(10*(R←INC)+F← LIM2, FRO,MT,
     (IF B[10*(R←INC)+F←LIM2]*SIGN(-MT)>0 THEN
ABS(B[10*(R←INC)+F←LIM2]) ELSE
IF ABS(PASS)=10*(R←INC)+F←LIM2 THEN
SIGN(PASS)*(K←K-3)  ELSE -1));
END; END;
END PROCEDURE PAWN;

  PROCEDURE ROOK;
BEGIN
INTEGER I,S;
FOR I← 1,-1,10,-10 DO
BEGIN
FOR S← FRO←I STEP I WHILE B[S]=0 DO
ENTER(S,FRO,MT,0);
IF B[S]<100 THEN
ENTER(S,FRO,MT,SIGN(-MT)*B[S]);
END;
END PROCEDURE ROOK;
!!
PROCEDURE KNIGHT;
BEGIN
INTEGER S;
FOR S← FRO←8,FRO-8,FRO←12,FRO-12
, FRO←19,FRO-19,FRO←21,FRO-21 DO
IF B[S]< 100 THEN
ENTER(S,FRO,MT,-(MT/3)*B[S]);
END OF KNIGHT;

PROCEDURE BISHOP;
```

```
BEGIN
INTEGER I,S;
FOR I← 9,11,-9,-11 DO
BEGIN
FOR S← FRO +I STEP I WHILE B[S]=0 DO
ENTER(S,FRO,MT,0);
IF B[S] <100 THEN
ENTER(S,FRO,MT,SIGN(-MT)*B[S]);
END;
END OF BISHOP;

PROCEDURE QUEEN;
BEGIN ROOK; BISHOP; END;

PROCEDURE KING;
BEGIN INTEGER RINC,FINC; INTEGER R,F;
INTEGER N,TEMP,TO.SQ;
FRO←KSQR[K];
FOR TOSQ←FRO+1,FRO-1,
FRO+10,FRO-10, FRO+9,FRO-9,
FRO +11, FRO-11 DO
IF B[TOSQ]<100 THEN
BEGIN
N←-SIGN(MT)*B[TOSQ];
IF ABS(FRO-TOSQ)=ABS(DIR) ^ CHECK[K]>0 ^B[TOSQ]=0 THEN
BEGIN N←-7;
END ELSE
IF N¬<0 THEN
BEGIN
H0←0;
TEST(5,TO.SQ,1,3-K);
IF H0>0 THEN
BEGIN
TEMP←-H0;
IF LIST[-H0]-LIST[1-H0]>1 THEN N←-1 ELSE
BEGIN H0←LIST[LIST[-H0]];
IF TABLE[3-K,3,H0] ≠-SIGN(MT) THEN N←-1 ELSE
IF ABS(TABLE[3-K,1,H0]-TABLE[3-K,2,H0])
  MOD 10 ≠0 THEN N←-1;
END;
LYST(TEMP,0);
END;
END;
ENTER(TOSQ,FRO,MT,N);
END;
IF CASTLE[K] THEN
BEGIN
IF CHECK[K]>0 THEN ! WE'RE IN CHECK SO
GO TO NEWCASTLE;  !  WE CAN'T CASTLE.
IF CASTLE[K-2] THEN    !! I CAN CASTLE LEFT
  BEGIN
   IF B[FRO-1]+B[FRO-2]=0 THEN
   BEGIN
     FOR FINC←FRO-1,FRO-2 DO
      BEGIN
```

```
            SEARCH(1,P[3-K],FINC,1,3-K);
             IF H0>0 THEN GOTO NO.CASTLE;
          END;
       ENTER(FRO-2,FRO,MT,0);
        END;
     END;
     NO.CASTLE:
     IF CASTLE[K+2] THEN  !!CHECK CASTLE RIGHT
     BEGIN
       IF B[FRO+1]+B[FRO+2]+B[FRO+3] =0 THEN
        BEGIN
          FOR FINC←FRO+1,FRO+2 DO
           BEGIN
           SEARCH(1,P[3-K],FINC,1,3-K);
              IF H0>0 THEN GOTO NEW.CASTLE;
           END;
       ENTER(FRO+2,FRO,MT,0);
        END;
     END;
     NEWCASTLE:
     END;
     END OF KING;

     PROCEDURE FIXB;
     BEGIN
     INTEGER I,J,T;
     I←TOSQ* .1-1;
     J←TOSQ- 10*I;
     RECORD.MOVE; ! IN B ARRAY
     IF B[TOSQ]≠0 ∨ ABS(MAN)=1 THEN B[-11]←B[150];
     IF ABS( MAN) =1 THEN
     BEGIN
       T←3-K;
       IF I=T*T*T THEN
       BEGIN
AL   IF CARD.MOVE THEN
AL   BEGIN LOGIC AC;
AL   NAME(AC);READ(<6C,1A>);
     MAN←MAN*(
     IF AC='R' THEN 2 ELSE
     IF AC='N' THEN 3 ELSE
     IF AC='B' THEN 4 ELSE 5);
AL   END ELSE MAN ← MAN * 5;
     END ELSE
     BEGIN IF B[TOSQ]=0 THEN
     BEGIN IF ABS(FRO-TOSQ) ≠10 THEN
     BEGIN
     B[10*(I+K+K-3)+J]←0;
     END;
     END;
     PASS← IF ABS(FRO- TOSQ)=V[6] THEN
      (3-K-K)*(FRO+10*(3-K-K))ELSE 0;
     END;
     END;
CO   TAKE CARE OF CASTLES..
```

```
AL IF CASTLE[K] THEN
   BEGIN
AL IF ABS(MAN)=2 THEN    ! WE ARE MOVING A ROOK;
AL BEGIN INTEGER T;
      T←K*K*K;  ! K-2= CASTLE LEFT SWITCH
   CASTLE[K+2]←CASTLE[K+2] ^10*T+8 ≠FRO      ;
   CASTLE[K-2]←CASTLE[K-2] ^10*T+1 ≠FRO      ;
      CASTLE[K]←CASTLE[K+2]^CASTLE[K-2];
AL END ELSE
      IF ABS(MAN)=6 THEN !!WE MOVED A KING
      BEGIN
         T←FRO-TOSQ;
         CASTLE[K]←FALSE;  !!NO MORE CASTLING FOR US
         IF  ABS(T)=2 THEN ! WE ARE CASTLING NOW
         BEGIN INTEGER T1;
            T1:=J*1.75-2.5; ! T1=1 OR 8.
     B[10*I+J+SIGN(T)]←B[10*I+T1];
     B[10*I+T1]←0; !! J+ SIGN(T) = 3 RO 5.
   B[-11]←B[120];  ! CASTLE IS IRREVERSIBLE MOVE
         END;
      END;
   END;
   B[TOSQ]←MAN;
   B[FRO]←0;
   END FIXB;

   INTEGER PROCEDURE DIRECTION(K,A);
   VALUE K,A; INTEGER K,A;
   BEGIN
   INTEGER M,S,INC;
   DIRECTION←0;
   FOR INC←9,11,10 DO
   IF ABS(K-A) MOD INC =0 THEN
   BEGIN
   DIRECTION ← INC*SIGN(A-K);
   GO BACK;
   END;
   IF ↓(K/10)=↓(A/10) THEN
   DIRECTION←SIGN(A-K);
   BACK:  END OF DIRECTION;

   PROCEDURE CHECKS;
   BEGIN INTEGER KG,K3,TEMP,MANT,Q,     TSQ,START,FINISH,I;
   IF CHECK[K]=2 THEN BEGIN
   ILLEGALIZE:
   Q←P[K];
   FOR I←1 STEP 1 UNTIL Q DO
   TABLE[K,4,I]←-25;
   GO AFTER; END;
   IF CHECK[K]=1 THEN BEGIN
   KG←KSQR[K];
   K3←3-K;
   SEARCH(1,P[K3],V[6],4,K3);  ! FIND CHECKING MOVBE
   TEMP←H0;
AL IF TEMP¬>0 THEN ERROR;!!TEMPORARY--FOR DEGUGGING
```

```
    MANT←ABS(TABLE[K3,3,TEMP]);
    H0←0;
    Q←TABLE[K3,2,TEMP];
    IF MANT=1∨MANT=3 THEN GO TO ILLEGALS;
    DIR←DIRECTION(KG,Q);
    H0←0;
    FOR TSQ←KG+DIR STEP DIR WHILE B[TSQ]=0 DO TEST(5,TSQ,1,K);
    ILLEGALS:
AL  TEST(5,Q,1,K);!!ALL THE LEGAL MOVES ARE NOW IN THE LIST
    IF H0¬>0 THEN GO ILLEGALIZE;
    START←LIST[1-H0];
    FINISH←LIST[-H0];
    FOR I← 1 STEP 1 UNTIL P[K] DO
    BEGIN
    FOR J← START STEP 1 UNTIL FINISH DO
    IF I= LIST[J] THEN GO TO LEGL;
    TABLE[K,4,I]←-25;
    LEGL:
    END;
CO  NOW WE ERASE THE LIST
    LYST( -H0,0);
    END;
    AFTER:
    END OF PROCEDURE CHECKS;

    PROCEDURE LISTCHECKS(ROUTINE,MT1,MT2);
    VALUE MT1,MT2;
    INTEGER MT1,MT2;
    PROCEDURE ROUTINE;
    BEGIN
    INTEGER T1,P1,K3,I,NAM,T2,D;
    P1←P[K]; NAM←H0;
    ROUTINE;  ! ENTER MOVES OF TYPE MT1 FROM KING'S SQUARE
    H0←0;
    FOR I ← P[K] STEP -1 UNTIL P1+1 DO
    IF TABLE[K,4,I] ¬< 0 THEN
    TEST(5,TABLE[K,1,I],1,K);  ! LIST OUR MOVES TO THIS SQUARE
    IF H0>0 THEN
    BEGIN
    FOR I← LIST[1-H0] STEP 1 UNTIL LIST[-H0] DO
    IF (T2←ABS(TABLE[K,3,LIST[I]]))=MT1∨T2=MT2 THEN
    BEGIN
    IF TABLE[K,4,LIST[I]]¬<0 THEN
    BEGIN
    LYST(NAM,LIST[I]);
    TABLE[K,4,LIST[I]]←-99; ! MARK IT SO ITS NOT COUNTED AGAIN
    END;
    END;
    LYST(-H0,0);
    END;
    P[K]←P1;
    H0←NAM;
    END OF LIST CHECKS;

    PROCEDURE CHECKERS;
```

```
CO CREATES A LIST NAMED H0 WITH ALL MOVES FOR SIDE K THAT
CO PUT SIDE 3-K INTO CHECK.
CO NOW IT ALSO PUTS -99  INTO THE VAL COLUMN OF THESE MOVES.
    BEGIN
    INTEGER T1,P1,K3,I,NAM,T2,D;
    K3←3-K;
    T1←FRO;  FRO←KSQR[K3];
    MT←(K3-K)*6;  NAM←DC[K];
    FOR I←LIST[1-NAM] STEP 1 UNTIL LIST[-NAM] DO TABLE[K,4,LIST[I]]←-99
    H0←NAM;
    LIST.CHECKS(ROOK,2,5);
    LIST.CHECKS(BISHOP,4,5);
    LIST.CHECKS(KNIGHT,3,0);
    NAM←H0;
CO NOW FOR PAWNS...
    D←K-K3; ! DIRECTION OF MOVE
    H0←0;
    FOR I←9,11 DO
    TEST(5, FRO←D*I,1,K);  ! LIST ATTACK SQUARES
    IF H0>0 THEN
    BEGIN
    FOR I←LIST[1-H0] STEP 1 UNTIL LIST[-H0] DO
    IF TABLE[K,3,LIST[I]]=-D ^ TABLE[K,4,LIST[I]] ¬< 0 THEN
    BEGIN
    LYST(NAM,LIST[I]);
    TABLE[K,4,LIST[I]]←-99;
    END;
    LYST(-H0,0);
    END;
    T1←IF K=1 THEN 70 ELSE 20;
     FOR I←1 STEP 1 UNTIL 8 DO
    IF B[T1+I]=-D THEN
    BEGIN
    SEARCH(1,P[K],  T1+I ,2,K);
    IF H0 >0 THEN
    BEGIN
    T2←TABLE[K,2,H0];
    FOR H0←H0,H0+1 WHILE TABLE[K,2,H0]=T2 DO
    IF TABLE[K,4,H0] ¬<0 THEN
    BEGIN
    LYST(NAM,H0);
    TABLE[K,4,H0]←-99;
    END;
    END ELSE ERROR;
    END;
    H0←NAM;
    FRO←T1;
    END OF CHECKERS;

    PROCEDURE THREATS;
CO CREATES A LIST OF-THREATS TO SQUARE FRO.
CO NOW IT ALSO PUTS -99  INTO THE VAL COLUMN OF THESE MOVES.
    BEGIN
    INTEGER T1,P1,K3,I,NAM,T2,D;
    K3←3-K;
```

```
    MT←(K3-K)*7;
    LIST.CHECKS(ROOK,2,5);
    LIST.CHECKS(BISHOP,4,5);
    LIST.CHECKS(KNIGHT,3,0);
    NAM←H0;
CO  NOW FOR PAWNS...
    D←K-K3; ! DIRECTION OF MOVE
    H0←0;
    FOR I←9,11 DO
    TEST(5, FRO←D*I,1,K);  ! LIST ATTACK SQUARES
    IF H0>0 THEN
    BEGIN
    FOR I←LIST[1-H0] STEP 1 UNTIL LIST[-H0] DO
    IF TABLE[K,3,LIST[I]]=-D ^ TABLE[K,4,LIST[I]] =< 0 THEN
    BEGIN
    LYST(NAM,LIST[I]);
    TABLE[K,4,LIST[I]]←-99;
    END;
    LYST(-H0,0);
    END;
    H0←NAM;
    END OF THREATS;

    PROCEDURE PINS;
    BEGIN
    INTEGER KS,D,SQ,PC,OUR.SIDE;
    BOOLEAN NO.ENCOUNTERS,DISC.CHECK;
    FOR K←1,2 DO
    BEGIN
    KS←KSQR[K]; OURSIDE←B[KS];  DC[3-K]←0;
    FOR D←1,9,10,11,-1,-10,-11,-9 DO
    BEGIN
    NO.ENCOUNTERS←TRUE;  DISC.CHECK←FALSE;
    FOR SQ←KS←D STEP D WHILE B[SQ]<100 DO
    BEGIN
    IF B[SQ]=0 THEN GO TO NEXT.SQUARE;
    IF NO.ENCOUNTERS THEN
    BEGIN
    IF B[SQ]*OURSIDE <0 THEN DISC.CHECK←TRUE;
    NO.ENCOUNTERS←FALSE;
    PC←SQ;
    END ELSE
    BEGIN
    IF B[SQ]*OURSIDE >0 THEN
    BEGIN
    DISC.CHECK←FALSE;
    GO TO NEXT.DIRECTION;
    END;
    IF ABS(B[SQ])=5 THEN GO TO PINNED;
    IF ABS(B[SQ])=2 ^(ABS(D)=1v ABS(D)=10) THEN GO TO PINNED;
    IF ABS(B[SQ])=4 ^(ABS(D)=9v ABS(D)=11) THEN GO TO PINNED;
    DISC.CHECK←FALSE;
    GO TO NEXT.DIRECTION;
    END;
    NEXT.SQUARE: END;
```

```
DISC.CHECK←FALSE;
GO TO NEXT.DIRECTION;
PINNED:
IF DISC.CHECK THEN
K←3-K;
SEARCH(1,P[K],PC,2,K);
IF H0>0 THEN
BEGIN
FOR T2←H0 STEP 1 WHILE TABLE[K,2,T2]=PC Do
BEGIN
IF TABLE[K,4,T2]=V[6] THEN GO TO NEXT.ROW;
IF ABS(D)=1 THEN
BEGIN IF
↓(TABLE[K,1,T2]*.1)=↓(TABLE[K,2,T2]*.1) THEN GO TO NEXT.ROW;
END ELSE
IF (TABLE[K,1,T2]-TABLE[K,2,T2]) MOD D =0 THEN GO TO NEXT.ROW;
IF DISC.CHECK THEN
BEGIN
IF TABLE[K,4,T2] ¬<0 THEN
LYST(DC[K],T2);
END
ELSE
TABLE[K,4,T2]←-12;  !-12 WILL MEAN ILLEGAL CAUSE OF PIN
NEXT.ROW:
END
END
ELSE
ERROR;
NEXT.DIRECTION:
IF DISC.CHECK THEN K←3-K;
END;
END;
END OF PINS;
!!
PROCEDURE FIXT;
BEGIN
        SWITCH IT← PA, RO,KN,BI,QU,KI;
BOOLEAN T1;
INTEGER T; T←FRO;
QSQR[2]←QSQR[1]←0;  ! ZERO QUEENS' SQUARES
P[2]←P[1]←0;
DIR←0;
FOR FRO ← 11 STEP 1 UNTIL 88 DO
IF B[FRO]¬=0 THEN
BEGIN
IF B[ FRO]<100 THEN
BEGIN
MT←B[FRO];
K←IF MT<0 THEN 2 ELSE 1;
GO TO IT[ABS(MT)];
PA:  PAWN; GO DNE;
RO:  ROOK; GO DNE;
KN:  KNIGHT; GO DNE;
BI:  BISHOP; GO DNE;
QU:  QUEEN;  QSQR[K]← FRO;  GO DNE;
```

```
    KI:  KSQR[K]←FRO;
    DNE:  END;
    END. NOW GENERATE KINGS SEPARATELY;
    FOR K←1,2 DO
    BEGIN
    COUNT(5,V[6],4,3-K);    ! COUNT CHECKS
    CHECK[K]←H0;
    IF H0>0 THEN CHECKS;
    END;
    P1←P[1];
    IF ABS(KSQR[1]-KSQR[2]) => 22 THEN
    BEGIN
    IF(ABS(((T2←KSQR[1]) MOD 10) - (T3←KSQR[2]) MOD 10) =2) ∨
      (ABS(↓(T2*.1)-↓(T3*.1))=2)THEN
    BEGIN
    T1←CASTLE[1];CASTLE[1]←FALSE; ! SAVE TIME IN KING THIS TIME
    K←1; MT←6; KING;
    CASTLE[1]←T1;  !RESTORE  CXASTLE SWITCH FOR KING
    END;
    END;
    K←2; MT←-6; KING;
    P[1]←P1;
    K←1; MT←6; KING;
    PINS;
    FOR K←1,2 DO
    DIRE[K,POINT+2]← DIRE [K,POINT+1]←P[K];
    FRO←T;  ! RESTORE FRO TO KEEP IT SAFE
AL  IF DEBUG =4 THEN ERROR;
AL  END OF ROUTINE TO CREATE TABLE FROM BOARD: FIXT;

    PROCEDURE PUSH.T;
      BEGIN
    INTEGER T; T←K;
    POINT←POINT+1;
    FOR K←1,2 DO
    BEGIN
    TABLE[K,1]←DIRE [K,POINT+1];
    FOR I←2,3,4 DO
    TABLE[ K,I]←TABLE[K,I-1]+ 1000;  ! NUMBER OF WORDS PER COLUMN
    END;
    FIXT;
    DIRE[1,0]←POINT;
    IF DIRE[1,POINT+2]-DIRE[1,1] > 999 THEN
    PRINT(<'ITS A BIG TABLE',E>);
    IF POINT>49 THEN PRINT(<'DEPTH =50,',E>);
    K←T;
    END OF PUSH DOWN TABLE ROUTINE;

    PROCEDURE UPDATE.B(H0,COLOR);
    VALUE H0, COLOR;
    INTEGER H0,COLOR;
    BEGIN
    INTEGER TEMP,I;
    INTEGER T; T←K;
AL  TEMP←0;
```

```
FOR I←4 STEP -1 UNTIL -1 DO
TEMP←10*TEMP+
SIGN(CASTLE[I]);
DIRE[3,POINT]←TEMP;
DIRE [4,POINT]←QSQR[2]+100*(QSQR[1]+10*(CHECK[2]+10*CHECK[1]));
DIRE[5,POINT]←B[120]+1000*(B[-11]-120);
MAN←TABLE[COLOR,3,H0];
TOSQ←TABLE[COLOR,1,H0];
FRO←TABLE[COLOR,2,H0];
K←COLOR;
FIXB;
PUSHT;
K←T;
END OF ROUTINE TO SET PARAMETERS AND CALL FIXB;

PROCEDURE T.FIXB;
  BEGIN
INTEGER T1,T2,T3;
INTEGER T; T←K;
  FOR I←1 STEP 1 UNTIL 8 DO
  FOR J←1 STEP 1 UNTIL 8 DO
B[10*I+J]←0;
T3←0;
FOR K←1,2 DO
BEGIN
FOR J←1 STEP 1 UNTIL P[K] DO
IF TABLE[K,2,J]≠T3 THEN
BEGIN
T3← TABLE[K,2,J];
B[T3]←TABLE[K,3,J];
END;
END;
K←T;
END OF FIX BOARD FROM TABLE ROUTINE;

PROCEDURE POP.T(LEVEL); VALUE LEVEL; INTEGER LEVEL;
BEGIN
INTEGER I,TEMP;
INTEGER T; T←K;
FOR K←1,2 DO
BEGIN
TABLE[K,1]←DIRE[K,LEVEL];
FOR I←2,3,4 DO
TABLE[ K,I]← TABLE[K,I-1]+1000;
P[K] ← DIRE[K,LEVEL+1]-DIRE[K,LEVEL];
KSQR[K]← TABLE [ K,2,P[K]]; ! RESET KING'S SQUARE
END;
POINT←LEVEL-1;
TFIXB;
TEMP←DIRE[3,POINT]; ! GET CASTLE INFORMATION
FOR I←-1 STEP 1 UNTIL 4 DO ! FROM DIRE. ONE DIGIT
CASTLE[I]←↓(TEMP*10↑-(I+1)) MOD 2>0; !PER CASTLE
K←T;
TEMP←DIRE[4,POINT];

QSQR[2]←TEMP MOD 100;
```

```
TEMP←TEMP*.001;
QSQR[1]←TEMP MOD 100;
TEMP←TEMP*.001;
CHECK[2]←TEMP MOD 10;
CHECK[1]←TEMP*.1;
B[120]←DIRE[5,POINT] MOD 1000;
B[-11]←DIRE[5,POINT]*.001+120;
END OF POP TABLE TO GIVEN LEVEL ROUTINE;

PROCEDURE LEGAL;
  BEGIN INIT← 1;
S←3-(K←IF MAN>0 THEN 1 ELSE 2);  IS IS NEXT SIDE TO MOVE
REPET: SEARCH(INIT,P[K],TOSQ,1,K);
       IF H0=0 THEN GO TO ILL;
IF TABLE[K,3,H0]= MAN
    ^TABLE[K,4,H0]¬<0
THEN BEGIN FRO←TABLE[K,2,H0]; GO OUT; END;
      INIT← H0+1; GO TO REPET;
ILL:  NAME(G[MAN],TOSQ);
PRINT(1);
       PRINT(<2A,' TO ',2D,' IS NOT A LEGAL MOVE. ',
             ' TRY AGAIN.',E>);
SUMMARIZE;
HERE←2;
IF SAVE ≠'P' THEN SAVE←'N';
GO TO MAKE.A.MOVE;
OUT:   END OF PROCEDURE  LEGAL;

PROCEDURE SETUP;
BEGIN INTEGER I,J;
FOR I←3 STEP 1 UNTIL 6 DO
FOR J←1 STEP 1 UNTIL 8 DO
B[10*I+J]←0;
B[18]←B[11]←2;
B[17]←B[12]←3;
B[16]←B[13]←4;
B[15]←5; B[14]←6;
FOR J←21 STEP 1 UNTIL 28 DO
BEGIN
B[50+J]←-(B[J]←1);
B[60+J]←-B[J-10];
END;
B[110]←0; ! MOVE COUNTER
B[-11]←  ! LAST IRREVERSIBLE MOVE
B[120]←120;  ! RECORDER MOVES
FOR I←-1 STEP 1 UNTIL 4 DO
CASTLE[I]←TRUE;
END OF SETUP STANDARD BOARD;
PROCEDURE DISCWRITE;
BEGIN END OF DISCWRITE;

PROCEDURE PAUSE;
BEGIN
INTEGER I;
LABEL EOF;
```

```
    PRINT(1);
    B[111]←S;  ! SIDE
    FOR I←-1 STEP 1 UNTIL 4 DO
    B[I+113]←CASTLE[I];
    B[89]←B[-11];  ! LAST IRREVERSIBLE MOVE
    B[ 90]← PASS;  ! EN PASSANT SWITCH;
    DISCWRITE(320,B[11],15,0,EOF);
    PRINT(<'PAUSED ON LFT 15',E>);
    HALT;
    EOF:
    PRINT(<'**CANNOT PAUSE ON LOGICAL FILE TYPE 15**',2E>);
    HALT;
    END OF PAUSE ONTO AND RECORD ROUTINE;
    PROCEDURE DISCREAD;
    BEGIN END OF DISCREAD;


    PROCEDURE UNPAUSE;
    BEGIN LABEL EOF;
    INTEGER I;
    DISCREAD( 320,B[11],15,0,EOF);
    NAME(↓((B[120]-119)*.55));
    PRINT(<'AFTER MOVE ',3D,
          ' LFT 15 HAS THE FOLLOWING BOARD',2E>);
    BOARDWRITE;
    FOR I←-1 STEP 1 UNTIL 4 DO
    CASTLE[I]← B[ I+113]¬=0;
    IF DEBUG>0 THEN J152;  ! PRINT CASTLE INFORMATION
    B[-11]←B[89];  ! RESTORE LAST IR MOVE
    PASS←B[90];
    B[89]←B[90]←100;  ! RESTORE BORDER
    FIXT;
    S←B[111];  ! SIDE
    K←S;
    PRINT(<P>);
    NAME(HERE);
    READ(<10C,1A>);
    HERE← IF HERE =0 THEN 2 ELSE 1;
    GO TO UN;
    EOF:
    PRINT(<'NO BOARD ON LOGICAL FILE TYPE 15',E>);
    SETUP;
    FIXT;
    END OF UNPAUSE;

    INTEGER PROCEDURE CAPVAL(ROW,K);
    VALUE ROW,K; INTEGER ROW,K;
    BEGIN
    INTEGER I,J;
AL  J←0;
AL  SEARCH(1,P[3-K],TABLE[K,1,ROW],2,3-K);
    IF H0>0 THEN
    BEGIN
    FOR I←H0 STEP 1 WHILE TABLE[3-K,2,I]=TABLE[K,1,ROW] DO
    J←MAX(J,TABLE[3-K,4,I]);
```

```
END;
ATAK :=J/3;
SEARCH(1,P[3-K], TABLE[K,1,ROW],1,3-K);
CAPVAL←TABLE[K,4,ROW];
IF CAPVAL = V[6] THEN
BEGIN
CAPVAL←2; ATAK←0;
END ELSE
IF H0>0 THEN
CAPVAL:=CAPVAL-V[ABS(TABLE[K,3,ROW])]+ATAK ELSE
CAPVAL:=CAPVAL+ATAK;
H0←CAPVAL;
END OF VALUE OF CAPTURE FUNCTION;

PROCEDURE CONSTANTS;
BEGIN
C[1,1]←2;  | NUMBER OF THREATS -1
C[2,1]←2;  | NUMBER OF THREATS
C[1,2]←.5;  |VALUE OF THREATS
C[2,2]←1; | VALUE OF THREATS
C[1,3]←.25; | MOBILITY
C[2,3]←.25; | MOBILITY
C[1,4]←1;  | CENTER CONTROL
C[2,4]←1;  | CENTER CONTROL
C[1,5]←1;  | NUMBER OF PINS
C[2,5]←1;  | NUMBER OF PINS
C[1,6]← 20;   | MATERIAL
C[2,6]←20;   | MATERIAL
C[1,7]←3;  |QUEEN PENALTY
C[2,7]←3;  | QUEEN PENALTY
C[1,8]←2;  | VALUE OF GOOD THREATS
C[2,8]←-10; | VALUE OF FORK
C[1,9]←3;  | NUMBER OF GOOD THREATS
C[2,9]←0;   | OPEN
C[1,10]←1; | CASTLE
C[2,10]←1; | CASTLE
C[1,11]←.75;|TABLE LENGTH
C[2,11]←.75;|TABLE LENGTH
C[1,12]←1;  | FUDGE FOR J154
C[2,12]←1;  | FUDGE FOR J154
CC[1]←1;  | CENTER CONTROL FACTOR--PAWN
CC[2]←.2; | ROOK
CC[3]←.9;  | KNIGHT
CC[4]←.4;  | BISHOP
CC[5]←.1;  | QUEEN
CC[6]←0;   | KING
X[1,0]←11;  | NUMBER OF FACTORS FOR US
X[2,0]←11; | NUMBER OF FACTORS FOR AN OPPONENT
END OF CONSTANTS;

PROCEDURE GREATEST(N,K);
  INTEGER N,K;
BEGIN INTEGER I, J, NAM, MAX1, ROW;
H0←0;
 NAM←0; TEST(3, 1, 4, K); IF H0>0 THEN
```

```
      BEGIN FOR I←1 STEP 1 UNTIL N DO
       BEGIN MAX1←0; FOR J←
       LIST [1-H0] STEP 1 UNTIL
       LIST [-H0] DO
       IF TABLE [K, 4, LIST[J]]> MAX1 THEN
        BEGIN ROW ←LIST [J];
        MAX1←TABLE [K, 4, ROW];
        END;
        IF MAX1=0 THEN BEGIN N←I-1; GO ERASE; END;
        LYST (NAM, ROW);
        TABLE [K, 4, ROW]←0;
      END;
     ERASE: LYST(-H0,0);
    END;
    H0←NAM:
    END OF GREATEST N;

    INTEGER
    PROCEDURE EVAL4(K);
    VALUE K; INTEGER K;
    BEGIN
    HALF TEMP;
    INTEGER K3,MAX1,MAX2,I,T1,T2,T3,T4;
    INTEGER NAM,MAX3;
    HALF DK;
    INTEGER ARRAY MAT,TOT[1:2];
    K3←3-K;
    FOR T2←1,2 DO
    FOR T1←1 STEP 1 UNTIL X[T2,0] DO
    X[T2.T1]←0;
    MAX1←
    T1←T2←T3←T4←
    0;
CO  MOBILITY
    COUNT(2,0,4,K);  ! LEGAL MOVES
    X[1,3]←H0; !OUR MOBILITY
    COUNT(2,0,4,K3);
    X[2,3]←H0;  !HIS MOBILITY
    IF H0=0 THEN
    BEGIN
    H0←EVAL4←IF CHECK[K3]>0 THEN 1000 ELSE 50ñ;
    GO TO THE
    END;
    H0←0;
     TEST(3,1,4,K3); !! GET ALL HIS CAPTURES
    IF H0>0 THEN
    BEGIN                                   !! FIND BEST ONE
    NAM←H0;
    FOR I←LIST[1-NAM] STEP 1 UNTIL LIST[-NAM] DO
    BEGIN
    TABLE[K3,4,LIST[I]]←CAPVAL(LIST[I],K3);
    IF H0>MAX1 THEN
CO  MAX1= OPPONENTS'S HIGHEST CAPTURE
    BEGIN
    MAX1←H0; MAX2←I; MAX3←ATAK;
```

```
    END;
    END;
    IF MAX1>0 THEN
CO  DELETE CAPTURED MAN'S THREATS
    BEGIN
    H0←0;
    TEST(5,TABLE[K3,1,LIST[MAX2]],2,K);
    IF H0>0 THEN
    BEGIN
    FOR I← LIST[1-H0] STEP 1 UNTIL LIST[-H0] DO
    TABLE[K,4,LIST[I]]←-1;
    MAX1←MAX1-MAX3;  | FUDGE OLD CAPVAL
    LYST(-H0,0);
    END;  |  H0¬>0 IF EN PASSANT MOVE.
    END ELSE MAX1←0;
CO  NOW I MAY AS WELL TAKE CARE OF OPPONENTS THREATS
    OPP:
    X[2,1]← LIST[-NAM]- LIST[1-NAM];  | NUMBER OF THRESTS -1
    FOR I←LIST[1-NAM] STEP 1 UNTIL LIST[-NAM]  DO
    T4←T4+ MAX(  0,TABLE[K3,4,LIST[I]]);
    X[2,2]←T4;  | VALUE OF THREATS >-U1
    LYST(-NAM,0);
    END;
    US: T3←T4←0;
CO  GET A LIST OF ALL OUR CAPTURES
    H0←0;
    TEST(3,1,4,K);
    IF H0>0 THEN
    BEGIN
    X[1,1]← LIST[-H0]-LIST[1-H0];   | NUMBER OF THREATS =1
    NAM←H0;
    FOR I←LIST[1-NAM] STEP 1 UNTIL LIST[-NAM] DO
    BEGIN
    T3←LIST[I];
    T4←T4+TABLE[K,4,T3];
    TEMP←TABLE[K,4,T3]←CAPVAL(T3,K);
    IF TEMP>0 THEN
    BEGIN
    T1←T1+1;
    T2←T2+TEMP;
    END;
    END;
    X[1,2]← T4;  ↑ VALUE OF THREATS
    X[1,9]←T1;  | NUMBER OF THREATS
    X[1,8]← T2;  | VALUE OF THREATS
CO  THESE THREATS HAVE POSITIVE VALUES ACCORDING TO CAPVAL
CO  SEE IF DOUBLE THREAT OR FORK
    IF T1>1 THEN
    BEGIN
    INTEGER N;
    N←2;
    GREATEST(N,K);
    IF N=2 THEN
    X[2,8]← TABLE[K,4,LIST[LIST[-H0]]]
    ELSE ERROR;  ↑ CAPVAL( SECOND G HIGHEST CAPTURE)
```

```
    LYST(-H0,0);
    END;
    LYST(-NAM,0);
    END;
CO  THE RECIPE CALLS FOR SOME MATERIAL BLENDED IN
    MAT[1]←MAT[2]←-V[6];!  REDUCED TO INCREASE SENSITIVITY.
    FOR I←1 STEP 1 UNTIL 8 DO
    FOR J←1 STEP 1 UNTIL 8 DO
    BEGIN
    T1←B[10*I+J];
    IF T1>0 THEN
    MAT[1]←MAT[1]+V[T1]
    ELSE IF T1<0 THEN
    MAT[2]← MAT[2]+ V[-T1];
    END;
    MAT[K]←MAX(0,MAT[K]-MAX1);
    X[1,6]← MAT[K];  ! OUT MATERIAL
    X[2,6]← MAT[K3];  ! HIS MATERIAL
    BEGIN
    INTEGER K,COL,SQ;
    HALF T5,T1;
    DK←(MAT[1]+MAT[2]) ↑2*.000164365549;  ! 1/(78)↑2
    FOR K←1,2 DO
    BEGIN T5←0;
CO  PINS
    COUNT(5,-12,4,3-K);
CO  ADD DISCOVERED CHECKS TO PINS.
    IF DC[K]>0 THEN
    BEGIN
    H0←H0+10*(LIST[-DC[K]]-LIST[1-DC[K]]+1);
    LYST(-DC[K],0);
    END;
    IF K=K3 THEN X[2,5]←H0 ELSE X[1,5]←H0;
CO  CENTER CONTROL
    T1← DK+DK;    ↑ N  2 POINTS PER SQUARE TIMES DECAYING FACTOR
    FOR SQ← 44,45,54,55,43,46,53,56 DO
    BEGIN
    IF SQ =43 THEN T1←DK;    ! OUTSIDE SQUARES GET 1 POINT TIMES DECAY F
    H0←0;
    TEST(5,SQ,1,K); ! LOOK FOR TOSQ'S ON SQ
    IF H0 > 0 THEN
    BEGIN
    FOR I← LIST[1-H0] STEP 1 UNTIL LIST[-H0] DO
    T5←T5+T1*CC[ABS(TABLE[K,3,LIST[I]])];
      LYST(-H0,0);
    END;
    IF K≠K3 THEN
    BEGIN
    IF B[SQ]>0 THEN
    X[K,4]←X[K,4]+T1*CC[B[SQ]]
    ELSE
    IF B[SQ]<0 THEN
    X[K3,4]←X[K3,4]+T1*    CC[-B[SQ]];
    END;
    END;
```

```
    IF K=K3 THEN
    BEGIN
    IF CASTLE[K] THEN
    BEGIN
    IF CASTLE[K-2] THEN X[2,10]←3*DK;  ! DECAYING BONUS FOR THE
    IF CASTLE[K+2] THEN X[2,10]←X[2,10]+2*DK;  ! ABILITY TO CASTLE LATE
    END;
    X[2,11]←P[K];  ! LENGTH OF MOVE TABLE
    X[2,4]←X[2,4]+T5;     END ! CENTER CONTROL
    ELSE
    BEGIN
    IF CASTLE[K] THEN
    BEGIN
    IF CASTLE[K-2] THEN X[1,10]←3*DK;  ! CASTLE SWITCH BONUS
    IF CASTLE[K+2] THEN X[1,10]←X[1,10]+2*DK;  ! LEFT AND RIGHT
    END;
    X[1,11]←P[K];              ! LENTGTH OF MOVE TABLE
    X[1,4]←X[1,4]+T5;     END;  ! CENTER CONTROL
    END ;
    END;
    K←3-K3;
    IF K=2 THEN
    BEGIN
    X[2,7]←DK*QP[9-↓(QSQR[2]*.1)];
    X[1,7]← DK*QP[↓(QSQR[1]*.1)];
    END ELSE
    BEGIN
    X[1,7]←DK*QP[9-↓(QSQR[2]*.1)];
    X[2,7]←DK*QP[↓(QSQR[1]*.1)];
    END;
    IF ABS(MAN) =6 THEN
    BEGIN IF ABS(FRO-TOSQ)=2 THEN
    X[1,10]←10; END; ! CASTLES ARE GOOD
    FOR K3← 1,2 DO
    BEGIN   T1←X[K3,0]; TEMP←0;
    FOR I← 1 STEP 1 UNTIL T1 DO
    TEMP← TEMP+ X[K3,I]* C[K3,I];
    X[K3,T1+1]← TEMP; ! TOTAL
    END;
    TEMP←X[1,X[1,0]+1]/(X[1,X[1,0]+1]+X[2,X[2,0]+1]);
    H0:= EVAL.4:=
    IF TEMP<.021 THEN 21 ELSE
    IF TEMP <.300 THEN 333.333*TEMP ELSE
    IF TEMP < .700 THEN 2000*TEMP-500 ELSE
    IF TEMP <.999 THEN 333.333*TEMP +666.667 ELSE 999;
    THE:
    IF DEBUG >1 THEN BEGIN
    J154;
    END;
    END OF EVAL3 AN EVALUATION FUNCTION EXTRAORDINAIRE;

SY PAGE
    PROCEDURE ORDER (N) BEST MOVES OF LIST H0 FOR SIDE: (N);
    VALUE N,K; INTEGER N,K;
    BEGIN
```

```
    INTEGER I,J,LOWER,TEMP,UPPER;
    INTEGER SAVE; SAVE←H0;
    LOWER←LIST[1-H0]; UPPER ←LIST[-H0];
CO  MAKE SURE IT'S ALL EVALUATED.
    FOR I←LOWER STEP 1 UNTIL UPPER DO
    IF TABLE[K,4,LIST[I]] ¬>20 THEN
    BEGIN
    UPDATE.B(LIST[I],K);
    IF REPEATS >1 THEN TEMP←500 ELSE
    TEMP←EVAL.4(K);
    POP.T(LEV);
    TABLE[K,4,LIST[I]]←TEMP;
    END;
CO  SORT N OF THEM IF THERE ARE THAT MANY.
    N← UPPER-MIN(UPPER-LOWER+1,N)+1;
    FOR J←UPPER STEP -1 UNTIL N DO
    FOR I←J-1 STEP -1 UNTIL LOWER DO
    IF TABLE[K,4,LIST[I]]>TABLE[K,4,LIST[J]] THEN ! SWITCH THEM
    BEGIN
    TEMP←LIST[J];
    LIST[J]←LIST[I];
    LIST[I]←TEMP;
    END;
    H0←SAVE; LIST[1-H0]←N;
    END OF ORDERING ROUTINE;

    PROCEDURE PICK.A.MOVE.5;
    COMMENT:  GIVEN THE CURRENT BOARD POSITION, THIS
    PROCEDURE WILL LOOK 'PICK' HALF MOVES DEEP TO FIND THE
    BEST MOVE FOR SIDE K.  IT WILL OUTPUT THE ROW NUMBER
    OF THE MOVE IT PICKS IN H0;
    BEGIN
    INTEGER I,VAL,K3;
    BOOLEAN ALL.MOVES;

    PROCEDURE SUGGEST;
    BEGIN ! PROCEDURE TO SUGGEST MOVES TO CONTINUE
CO  EXPLORING TO DEEPER LEVELS.  OUTPUT LIST←NAM[LEV];
    H0←0;
    IF LEV > PICK THEN  ! WE'VE GONE DEEP ENOUGH.
    BEGIN  NAM[LEV]←0;
           GO TO EXIT;
    END;
    IF LEV =1 THEN
    BEGIN
    CHECKERS;
    TEST(3,1,4,K);  ! ADD CAPS TO LIST OF CHECKS.
    ORDER(N,K);
    NAM[LEV]←H0;
    IF H0>0 THEN I←1+LIST[-H0]-LIST[1-H0] ELSE I←0;
    H0←0;
    TEST(5,0,4,K);  ! QUIET MOVES.
    IF H0 >0 THEN
    BEGIN
    ORDER(MAX(2,N-I),K);
```

```
    IF I>0 THEN
    BEGIN
    FOR I←LIST[1-NAM[LEV]] STEP 1 UNTIL LIST[-NAM[LEV]] DO
    LYST(H0,LIST[I]);
    LYST(-NAM[LEV],0);
    END;
    END ELSE H0←NAM[LEV];
    IF H0=0 THEN ! NO MOVES.
    BEGIN
    REASON←IF CHECK[K]>0 THEN 0 ELSE 500; !CHECKMATE OR STALEMATE.
    GO TO EXIT;
    END;
    ROW[1]←LIST[LIST[-H0]];  ! IN CASE TIME LIMIT EXCEEDED
    ALL.MOVES←TRUE;
    ORDER(N←2,K);
    IF PICK >1 THEN
CO  CREATE A KILLER LIST FOR OPPONENT.
    BEGIN
    KL←0;
    NAM[LEV]←H0;
    H0←0;
    K←
    K3←3-K;
    CHECKERS;  ! CREATEMA LIST OFCHECKERS SO THEY'RE NOT ON LIST
    IF H0>0 THEN LYST(-H0,0);
    H0←0;
    K←3-K3;
    TEST(5,0,4,K3); ! HIS QUIET MOVES
    IF H0>0 THEN
    BEGIN
    ORDER(N,K3);
    FOR I←LIST[1-H0] STEP 1 UNTIL LIST[-H0] DO
    BEGIN
    LYST(KL,TABLE[K3,2,LIST[I]]);
    LYST(KL,TABLE[K3,1,LIST[I]]);
    END;
    LYST(-H0,0);  ! ERASE ORIGINAL LIST
    END;
    H0←NAM[LEV];
    END;  ! FIRST PART OF KILLER HEURISTIC
    GO TO EXIT;
    END ELSE
    IF CHECK[K]>0 THEN
    BEGIN
    TEST(2,0,4,K);     ! CREAT A LIST OF ALL LEGAL MOVES
    ALL.MOVES←TRUE;
    END ELSE
    BEGIN
    CHECKERS;
    TEST(3,1,4,K);
    ALL.MOVES←FALSE;
    IF LEV=2 THEN
    BEGIN
    NAM[LEV]←H0;
    IF KL>0 THEN
```

```
    BEGIN
    IF DEBUG >2 THEN J151(KL);
    FOR I←LIST[1-KL] STEP 2 UNTIL LIST[-KL] DO
    BEGIN
    SEARCH(1,P[K],LIST[I],2,K);
    IF H0>0 THEN
    BEGIN
    SEE:
    IF TABLE[K,1,H0]=LIST[I+1] THEN
    BEGIN
    IF TABLE[K,4,H0]=0 THEN LYST(NAM[LEV],H0);
    END ELSE
    BEGIN
    H0←H0+1;
    IF TABLE[K,2,H0]=LIST[I] THEN GO TO SEE;
    END;
    END;
    END;
    END;
    H0←NAM[LEV];
    FRO←TOSQ;
    IF ABS(MAN) ≠6 THEN THREATS;
    END;
    END;
    ORDER(N,K);  ! LOOK AT THE BEST N IN ORDER.
    EXIT:
    NAM[LEV]←H0;
AL  IF DC[2] ≠H0 ^DC[2]>0 THEN LYST(-DC[2],0);
AL  IF DC[1] ≠H0 ^DC[1]>0 THEN LYST(-DC[1],0);
    IF DEBUG >2^H0>0 THEN J153(H0,K);
    END OF SUGGEST;

    PROCEDURE  DESCEND;
    BEGIN
    VAL←TABLE[K,4,LIST[LIST[-NAM[LEV]]]];
    UPDATE.B(LIST[LIST[-NAM[LEV]]],K);
    IF DEBUG >0 THEN
    BEGIN !
    NAME(G[MAN],TOSQ,VAL);
    PRINT(←$5*LEV$<'.'>,<2A ,'→',2D.,1C,4D.,E>);
    END;
    LEV←LEV+1;
    K←3-K;  ! SWITCH SIDES.
    END;  ! OF DESCEND A LEVEL ROUTINE.
    ALPHA[1]←-1;
    LEV←1;
    IF DEBUG>1 THEN
    BEGIN
    PRINT(<P>);
    PRINT(<E,50C,'EVALUATION FUNCTION',E,
    'MOVE',8C,'NO THTS V THTS   MOBIL  ',
    'CENT CON PIN→DOS   MAT   QUEEN  FORKS  ',
    '  G THTS  CASTLE   DEVEL   TOTAL ',
    2E>);
    END;
```

```
    LOOP:
    IF REPEATS >1 THEN
    BEGIN
    VAL←500; GO TO ASCEND;
    END;
    SUGGEST;  ! MOVES TO CONSIDER DEEPER.
    IF NAM[LEV]>0 THEN
    BEGIN
    IF ¬ ALL.MOVES THEN
    BEGIN
    ALPHA[LEV]←1000-VAL;
    IF 1000-ALPHA[LEV] ¬>ALPHA[LEV-1] THEN GO TO HIGHER.LEVEL; ! ALPHA-
    END;
      DESCEND;  ! TO NEXT HALF MOVE LEVEL AND
      GO TO LOOP;
    END;
    ASCEND:   LEV←LEV-1;
    IF LEV =0 THEN ! WE'RE DONE.
    BEGIN
      REASON←1000-VAL;  ! RELATIVE VALUE
      HO←ROW[1];  ! ROW NUMBER IN TABLE OF MOVE.
    IF REASON =0 THEN
    BEGIN
    N←50; PICK←2; LEV←1; GO TO LOOP;
    END;
      GO TO EXIT;
    END;
    POP.T(LEV);  !BRING BACK THE TABLE AND BOARD.
    K←3-K;  ! SWITCH SIDES.
    IF 1000-VAL ¬> ALPHA[LEV-1] THEN ! SKIP IT.(ALPHA-BETA HEURISTIC)
    BEGIN ALPHA[LEV]←999;
AL  IF DEBUG > 2 THEN
    BEGIN NAME(LEV,1000-VAL,ALPHA[LEV-1]);
          PRINT(<2D.,-4D,' ¬>',-4D,' ASCENDING.',E>);
    END;
    GO TO HIGHER.LEVEL;
    END;
    IF VAL > ALPHA[LEV] THEN !WE FOUND A BETTER MOVE.
    BEGIN ! REMEMBER IT.
      ALPHA[LEV]←VAL;
      ROW[LEV]←LIST[LIST[-NAM[LEV]]];
AL  IF DEBUG>2 THEN
    BEGIN NAME(LEV,VAL);
        PRINT(<2D.,' NEW ALPHA[LEV] =',-4D,E>);
AL  END;
    END;
    GO.ON:
    IF LIST[-NAM[LEV]]>LIST[1-NAM[LEV]] THEN ! THERE ARE MORE MOVES.
    BEGIN
    IF ALPHA[LEV]=1000 THEN GO TO HIGHER.LEVEL;
    LIST[-NAM[LEV]]←LIST[-NAM[LEV]]-1;
    DESCEND;
    GO TO LOOP;
    END;
    HIGHER.LEVEL:  VAL←1000-ALPHA[LEV];
```

```
    ALPHA[LEV]←-1;
    LYST(-NAM[LEV],0); !ERASE THIS LEVEL'S LIST OF MOVES.
    GO TO ASCEND;
    EXIT:
    IF KL >0 THEN
    BEGIN
    LYST(-KL,0); !!  ERASE KILLER LIST
    KL←0;
    END;
    END OF PICK A MOVE 5;
```

```
CO EXECUTION STARTS HERE↓↓↓↓↓↓↓.
    PRINT(< E,15C,
    'RUBENFELD CHESS PROGRAM.  5/B. MAY 1968.'
    ,E>);
    S←2; HERE←1;          ! WHITE MOVES FIRST PLAYING AN OPPONENT
    N←5;                ! STANDARD TREE=5 MOVES ACROSS
    DEBUG←0; PICK←3;  !  STANDARD VALUES
    G[-6]←'BK';
    G[-5]←'BQ';
    G[-4]←'BB';
    G[-3]←'BN';
    G[-2]←'BR';
    G[-1]←'BP';
    G[0]←'  ';
    G[1]←'WP';
    G[2]←'WR';
    G[3]←'WN';
    G[4]←'WB';
    G[5]←'WQ';
    G[6]←'WK';
    G1[0]←'  ';
    G1[1]←'//';
    LIST[0]←0;
    POINT←0;
    DIRE[1,1]←TABLE[1,1];
    DIRE[2,1]←TABLE[2,1];
    QP[9]←QP[0]←0;
    QP[1]←0;  QP[2]←1;  QP[3]←3; QP[4]←6;
    QP[5]←8;  QP[6]←1;  QP[7]←QP[8]←0;!!QUEEN PENALTIES
    V[6]←16; V[5]←9 ; V[4]←3;
    V[3]←3; V[2]←5; V[1]←1; V[0]←0;
    V[7]←25;
    FOR I←0 STEP 1 UNTIL 30 DO ALPHA[I]←-1;
    FOR I←-10 STEP 1 UNTIL 109 DO
    B[I]←100;
    B[-11]←  ! LAST IRREVERSIBLE MOVE
    B[120]←120;
    CONSTANTS; ! FILL THE CONSTANTS FOR PICKAMOVE

    GO PAST;
    ERR:
    ERROR; SUMMARIZE;BOARDWRITE;DIRECTORY;HALT;
    GUN:  POPT(1);  !  ASCEND TO ORIGINAL LEVEL FOR BOARD
    IF HERE =2 THEN
```

```
    BEGIN
    IF SAVE ≠'N' THEN
    PAUSE;
    END;
    PRINT(<E,                          'BEST MOVE SO FAR FOLLOWS:',E>);
    IF ROW[1]<0 THEN
    BEGIN
    BOARDWRITE;
    IF SAVE ≠'N' THEN
    PAUSE;
    END;
    K←3-S;
    H0←ROW[1];
    REASON←MAX(ALPHA[1],100);  ! DON'T RESIGN..NED MORE TIME.
    GO TO GUM;
    PAST:
    NAME(WORB);
    READ(<W,1A>);
    NAME(T1,T2);
    READ(<12C,1A,14C,1A>);
    IF T1> 32 THEN PICK ← T1-32;
    IF T2>32 THEN DEBUG←T2-32;
    NAME(T3);
    READ(<16C,1A>);
    IF T3>32 THEN N←T3-32;
    NAME(SAVE);
    READ(<18C,1A>);
    NAME(T1,T2);
    READ(<20C,1A,1A>);
    RPT ←MAX(T1-32,0);
    IF T2 ¬> '␣' ∧T2¬< '0' THEN RPT ←RPT *10←T2-32;
    IF WORB ='P' THEN UNPAUSE
    ELSE IF WORB='R' THEN
    BEGIN
    COMMENT:  THE FOLLOWING CONTROL CARD IS EXPECTED
    12345678901234567890123456789012345678 90
    READ THE FOLLOWING BOARD.  TFTFTF 10;
    NAME($FOR I←-1 STEP 1 UNTIL 4 DO$
        (CASTLE[I]),B[120]);
    READ(<28C>,→6<1T>,<3D>);
    B[-11]←
    B[120]←2*B[120]+120;
    NAME( I→8(J→8(B[10*I+J])));
    READ(<W, 64F>);
    END ELSE SETUP;
    PRINT(<2E>);
    BOARDWRITE;
    FIXT;
    IF RPT>0 THEN GO TO UN;
CO  INSERT TESTS HER↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
    PRINT(1);
    PRINT(<'CHOOSE YOUR SIDE. (BLACK OR WHITE ',
    'STARTING IN COLUMN 1.',')',E>);
    NAME(WORB);
    READ(<W,1A>);
```

```
PRINT(0);
IF WORB='B' THEN
BEGIN
S←HERE←1;
GO TO MAKE.A.MOVE;
END
ELSE IF WORB≠'W' THEN
BEGIN
  HERE←2;
S←1;
GO MAKE.A.MOVE;
END ELSE
S←HERE←1;
REEDAMOVE:      BEGIN
PRINT(1);
PRINT(<'YOUR MOVE',E>);
IF SAVE ='N' THEN
BEGIN PRINT(<'NO PAUSE ON LFT 15',E>);
HALT;
END;
IF B[120]>120 THEN
PAUSE;
UN:
K←S←3-S;
PRINT(1);
NAME(WORB, TOSQ);
READ(<W,2A,1B,2D>);
PRINT(0);
FOR I←-6 STEP 1 UNTIL 6 DO
IF G[I]=WORB THEN
BEGIN MAN←I;  GO TO WON;
END;
NAME(FRO); READ(<1C,2D>);
FOR K←1,2 DO
BEGIN INIT←1;
  REPET:
  SEARCH(INIT,P[K],FRO,2,K);
  IF H0>0 THEN
    BEGIN
IF TABLE[K,1,H0]= TOSQ ^TABLE[K,4,H0] ¬<0 THEN
BEGIN MAN←TABLE[K,3,H0];
      GO FIXBEE;
END ELSE
      BEGIN INIT←INIT+1; GO REPET; END;
    END;
  END;
RDER:
PRINT(1);
PRINT(<'***ILLEGAL MOVE***',2E>);
PRINT(0);
S←(B[120] MOD 2) +1;  !SET S TI CORRECT SIDE TO MOVE
SUMMARIZE;
IF SAVE ≠'P' THEN SAVE ←'N';  ! DON'T PAUSE
HERE←2; GO MAKEAMOVE;
WON:
```

```
END OF READ A MOVE;
LEGAL;
FIXBEE:
CARD.MOVE←TRUE;
FIXB;
CARD.MOVE←FALSE;
BOARDWRITE;
IF REPEATS>1 THEN
BEGIN
PRINT(<E,'↓↓ POSITION REPEATED 3 TIMES.  ↑TALEMATE. ↓↓',2E>);
SUMMARIZE;
HALT;
END;
FIXT;
IF DEBUG =5 THEN EVAL.4(3-S);
IF RPT>0 THEN
BEGIN RPT←RPT-1;
PRINT(<P>);
      GO TO UN;
END;

MAKE.A.MOVE:
K←S;
S←3-S;
PICK.A.MOVE.5;
GUM:
PRINT(1);
PRINT(<P>);
IF H0=0 THEN
BEGIN  ! NO MOVE.
IF REASON =0 THEN
PRINT(<E,'↓↓ CHECKMATED ↓↓',E>) ELSE
PRINT(<E,'→→STALEMATED ←←',E>);
SUMMARIZE;
HALT;
END;
MAN←TABLE[K,3,H0];
FRO←TABLE[K,2,H0];
TOSQ←TABLE[K,1,H0];
FIXB;  ! MAKE THE MOVE ON THE BOARD
ROW[1]←-1;
FIXT;  ! MAKE THE MOVE TABLE
IF REASON =1000 THEN ! WE WON↑↑↑↑↑↑↑↑↑↑↑↑↑
PRINT(<'↑↑↑↑↑↑↑ CHECKMATE ↑↑↑↑↑↑↑↑↑',2E>)
ELSE IF CHECK[3-K]>0 THEN
PRINT(<'↑↑ CHECK ↑↑',E>);
NAME(↓((B[120]-118.75)*.5));
PRINT(<3D.,2B>);
IF MAN > 0 THEN PRINT(<'WHITE '>)
      ELSE PRINT(<'BLACK '>);
NAME(G[MAN],TOSQ,REASON);
PRINT(<'MOVES ',2A,' TO ',2D,' WITH RELATIVE VALUE ',4D,2E>);
PRINT(TTY);
BOARDWRITE;
IF REASON =1000 THEN
```

```
    BEGIN
    SUMMARIZE;
    HALT;
    END;
    IF REPEATS>1 THEN
    BEGIN
    PRINT(<E,'POSITION REPEATED 3 TIMES. STALEMATE↑',2E>);
    SUMMARIZE;
    HALT;
    END;
    IF REASON <100 ^REASON ≠0 ^HERE =1 THEN
    BEGIN
    PRINT(<'I FEAR ',Q,'TIS FUTILE TO CONTINUE.  UNLESS YOU WISH TO ',
    'PROLONG THE AGONY, I RESIGN.',2E>);
    SUMMARIZE;
    END;
    FOR I←-2 STEP -2 UNTIL -100 DO
    IF LIST[I] ≠ 0 THEN
    BEGIN
AL  IF DEBUG > 1 THEN
    J151(-I);
    LYST(I,0);
    END;
    PRINT(0);
    GO TO WHERE[HERE];
    END OF CHESS PROGRAM;
    BEGIN LIBRARY PROCEDURE LINK;LINK(5)END
```