Algorithms supplement

Algorithm 89

A PROGRAM TO REFEREE KRIEGSPIEL AND CHESS

T. J. Buckholtz and C. S. Wetherell
Lawrence Radiation Laboratory
PO Box 808
L-71
Livermore, California 94550, USA

Author's note:
The program given here is a referee for the game of *Kriegspiel.* It was originally designed as the heart of a Kriegspiel monitor system (Wetherell, Buckholtz, and Booth, 1972), but may be used as well to arbitrate chess. Because Kriegspiel requires more information from a referee than does chess, the program might be shortened for a chess application. However, the extra information might well be useful even in non-Kriegspiel situations. This program compares favourably with that of Bell (1970). Readers unfamiliar with the game of Kriegspiel may wish to consult Wetherell *et al.* (1972).

(1, 8)	(2, 8)	(3, 8)	(4, 8)	(5, 8)	(6, 8)	(7, 8)	(8, 8)
(1, 7)	(2, 7)	(3, 7)	(4, 7)	(5, 7)	(6, 7)	(7, 7)	(8, 7)
(1, 6)	(2, 6)	(3, 6)	(4, 6)	(5, 6)	(6, 6)	(7, 6)	(8, 6)
(1, 5)	(2, 5)	(3, 5)	(4, 5)	(5, 5)	(6, 5)	(7, 5)	(8, 5)
(1, 4)	(2, 4)	(3, 4)	(4, 4)	(5, 4)	(6, 4)	(7, 4)	(8, 4)
(1, 3)	(2, 3)	(3, 3)	(4, 3)	(5, 3)	(6, 3)	(7, 3)	(8, 3)
(1, 2)	(2, 2)	(3, 2)	(4, 2)	(5, 2)	(6, 2)	(7, 2)	(8, 2)
(1, 1)	(2, 1)	(3, 1)	(4, 1)	(5, 1)	(6, 1)	(7, 1)	(8, 1)

The Co-ordinates on the Board

Piece	*Code*	*Starting Square*
White pseudo-piece	7	None
White rook	6	(1, 1), (8, 1)
White queen	5	(4, 1)
White bishop	4	(3, 1), (6, 1)
White knight	3	(2, 1), (7, 1)
White pawn	2	(x, 2), x = 1, 8
White king	1	(5, 1)
Black king	−1	(5, 8)
Black pawn	−2	(x, 7), x = 1, 8
Black knight	−3	(2, 8), (7, 8)
Black bishop	−4	(3, 8), (6, 8)
Black queen	−5	(4, 8)
Black rook	−6	(1, 8), (8, 8)
Black pseudo-piece	−7	None
Empty cell	0	

The Pieces

Fig. 1 Representation of the board

Use of the Referee
Within the referee, a special algebraic notation is used for the chess board and pieces. **Fig. 1** shows the co-ordinates for each of the squares and the numerical codes for the men. Note the pseudo-piece with value seven and the use of positive values for white men and negative for black. A move is specified as a quintuple: the co-ordinates for the initial square of the move, the co-ordinates of the destination square, and, if a promotion is possible, the absolute value of the code for the piece promoted to. A castling move is made by simply moving the king two squares in the appropriate direction. This coding method is similar to that used for international telegraphic communication of chess.

The referee is written in FORTRAN as a set of seven subroutines. The subroutine KSREF is the entry to this set. The user provides input through the common block KSINP and receives results through KSINP and the common block KSOUT. Common block KSINP consists of the eight element vector IDATA, which controls the referee's actions, and the 8 × 8 × 2 array IBOARD, which contains a representation of the chess board. A 30 × 2 array MINFO, which returns information about an attempted move, and a 50 element vector NOS, which summarises past illegal attempts, comprise common block KSOUT. The vector IDATA is used as working storage by KSREF. KSREF maintains the other three arguments as historical data providing continuity between calls and the calling routine should treat them as read-only data. The use of arguments in common provides rapid argument transmission and allows equivalenced mnemonic names for some specific elements of the arguments. **Tables 1** through **4** describe the coding for argument values. All arguments are typed integer.

Details of the arguments
The vector IDATA describes an attempted move. The elements of IDATA are IRULES, ICOLOR, IFROMH, IFROMV, ITOH, ITOV, IPROMO, and ITRYRQ. When KSREF is called with ICOLOR set to zero, the arrays IBOARD, MINFO, and NOS are filled with initial values. When KSREF is called with IRULES set to 2 (indicating Eastern rules) and ITRYRQ set to 1, current pawn try data is reported. All other calls to KSREF must have ITRYRQ set to zero.

In all other cases, IDATA describes a normal chess move. IRULES specifies the Kriegspiel rules variation in effect and ICOLOR the player with the move. The pair (IFROMH, IFROMV) is the co-ordinates of the initial square of the move and the pair (ITOH, ITOV) the co-ordinates of the destination square. IPROMO specifies the replacement piece if a promotion is to take place. If the attempt was a legal move, ICOLOR changes sign. If the attempt ends the game, ICOLOR is set to zero. IRULES is returned unchanged and the other elements of IDATA may be destroyed.

IBOARD reports and maintains the game board. The subarray IBOARD(*, *, 1) is the chess board with numerical codes for the men. The subarray IBOARD(*, *, 2) contains historical information for the control of castling and *en passant* pawn captures. Initialisation causes the men to be placed in their starting positions. After each legal move IBOARD is changed to reflect the move.

MINFO is a table giving a report on an attempted move. The rules set in use is reported in MRULES. MTERMN, MLOST, MSQRH, MSQRV, MCAPT, and MPROMO detail the last legal move. MTERMN reports a game termination; MLOST, MSQRH, MSQRV, and MCAPT announce a capture; and MPROMO reports a pawn promotion. If the last attempt was not legal, MATMPT is set appropriately.

MCOLOR tells the player with the move. MCHEKA and MCHEKB record current checks. MTRYEA returns try request answers for Eastern rules. MNOS contains the number of attempts receiving a 'no' since the turn began. Pawn try information fills out the table. The numbers of single and double tries are contained in MSTRY and MDTRY, respectively. The appropriate co-ordinates are contained in the subarrays described in **Table 3.** Pawn try co-ordinates are not recorded under Eastern rules.

The vector NOS is 50 elements long. Since a repeat of an attempt previously declared a 'no' during a turn is a 'heck-no', KSREF maintains a list of illegal attempts in NOS. The first element specifies the number of 'no' attempts presently recorded and the rest of the elements are codes for the illegal attempts. A legal move always causes NOS to be cleared. If more than 49 illegal moves occur in one

Table 1 Explanation of input vector IDATA

Information in the vector IDATA is supplied by the program calling KSREF and specifies the type of computation to be performed. There are three cases:

I. Initialisation of the game.
A. Processing of a move attempt.
T. Processing of a pawn try request (Eastern rules).

A value must be specified for each variable designated as belonging to a given case. Only ICOLOR and IRULES are returned meaningfully.

Subscript	*Name*	*Cases*	*Values*	*Meaning*
1	IRULES	I, A	1	Western rules.
		I, A, T	2	Eastern rules. IRULES is returned unchanged.
2	ICOLOR	I	0	Begin the game.
		A, T	1	White to move.
		A, T	−1	Black to move. ICOLOR is returned with value 1 if called with value 0; with sign reversed if the move was legal; with value 0 if the move ends the game.
3	IFROMH	A	1 to 8	The pair (IFROMH, IFROMV) is the
4	IFROMV	A	1 to 8	co-ordinates of the initial square of the move.
5	ITOH	A	1 to 8	The pair (ITOH, ITOV) is
6	ITOV	A	1 to 8	the co-ordinates of the destination square.
7	IPROMO	A	0	Zero if no promotion.
		A	3 to 6	Absolute valve of code for promotion piece.
8	ITRYRQ	A	0	No meaning.
		T	1	Return pawn try response for player ICOLOR if IRULES = 2.

Table 2 Explanation of the input array IBOARD

The first two co-ordinates of IBOARD give a square location in Cartesian co-ordinates.

The value of IBOARD(i, j, 1) is

a: if square (i, j) is unoccupied, 0.
b: if square (i, j) is occupied,
  1: the absolute value is given by the code
    1 = king
    2 = pawn
    3 = knight
    4 = bishop
    5 = queen
    6 = rook.
  2: the sign is given by the colour of the man on the square, where
    + = white
    − = black.

The value of IBOARD(i, j, 2) is

a: if square (i, j) is occupied by a rook which has moved or which was created by a promotion, 1.
b: if there is a king on square (i, j), twice the number of times the king is in check plus one if the king has ever moved. (Under Eastern rules, double checks are not noted.)
c: if square (i, j) contains a pawn which moved two squares on the last move, 1.
d: if square (i, j) contains a pawn which had moved before the last move or which moved one square on the last legal move, 2.
e: otherwise, 0.

Table 3 Explanation of the output array MINFO

Entry	*Name*	*Notes*	*Values**	*Meanings**
(1, 1)	MTERMN		1	Game in progress.
			2	Checkmate.
			3	Stalemate.
(1, 2)	MRULES	r	1	Western rules.^{w}
			2	Eastern rules.^{e}
(2, 1)	MLOST	a	2-6	Code for piece if any captured on last legal move.
(3, 1)	MSQRH	a	1-8	If a man was captured on the
(3, 2)	MSQRV	a	1-8	last move, the pair (MSQRH, MSQRV) is the co-ordinates from which the captured man was removed.
(4, 1)	MCAPT	b, r	1	A piece was captured on the last move.^{w}
			2	A pawn was captured on the last move.^{w}
			3	A man was captured on the last move.^{e}
(5, 1)	MPROMO	a, p	1	Pawn promoted on last move.
(6, 1)	MATMPT		1	Last attempt was a 'no'.
			2	Last attempt was a 'heck-no'.
(6, 2)	MCOLOR		1	White to move.
			2	Black to move. If MTERMN is greater than one, the player to move is mated.
(7, 1)	MCHEKA	r, c	1	Check exists.^{e}
			2	Long diagonal check.^{w}
			3	Short diagonal check.^{w}
			4	Vertical check.^{w}
			5	Horizontal check.^{w}
			6	Knight check.^{w}
(7, 2)	MCHEKB	r, c^{w}	2-6	Codes same as MCHEKA. MCHEKB is set for double checks under Western rules.
(8, 1)	MTRYEA	r, i^{e}	1	Pawn tries requested; none.^{e}
			2	Pawn tries requested; some.^{e}
(8, 2)	MNOS		0-∞	Number of 'no' responses since last legal move.
(9, 1)	MSTRY	r, i	0-14	Number of single pawn tries available.^{w}
			1	Some pawn tries exist.^{e}
(k+9, 1) $1 \leq k \leq 14$ (k+9, 2)		r, i^{w}	1-8	Under Western rules, destination co-ordinates of all available single pawn tries for player MCOLOR.
(24, 1)	MDTRY	r^{w}	0-6	Number of double pawn tries available.^{w}
(k+24, 1) $1 \leq k \leq 6$ (k+24, 2)		r^{w}	1-8	Under Western rules, destination co-ordinates of all available double pawn tries for player MCOLOR.

Notes

* 0 is a possible value for any entry in MINFO except MTERMN, MRULES, and MCOLOR (*e.g.*, MLOST = 0 is there was no legal capture on the last legal move).
a Information given only to the player with the move.
b Information given only to the player without the move.
c MCHEKA reports the first check. If, under Western rules, a double check appears, MCHEKB reports the second check.
i Used as input to KSREF. Must not be disturbed.
p The authors recommend ignoring this variable.
r The value depends on the rules set in use.
  . . .^{e} = pertains to Eastern rules only.
  . . .^{w} = pertains to Western rules only.

turn, later ones are simply not entered in NOS. NOS is generally of little interest to the calling program.

Description of the code

This section outlines the purposes of the various sections of the referee code and presents a verbal flow. Details may be found in the comments in the code listing. **Fig. 2** is a block flowchart for the subprograms.

Seven functions and subroutines comprise the referee, which is

Table 4 Explanation of the output vector NOS

Element	*Value*
NOS(1)	1 + the number of attempted moves resulting in a return of 'no' since the last legal move. NOS(1) is less than 51.
NOS(i)	1 is less than i and i is less than or equal to NOS(1). A table of coded move attempts which have resulted in announcements of 'no'. The code is 729p + 81q + 9r + s where the move attempted was (p, q) to (r, s).
NOS(i)	NOS(1) is less than i. Undefined.

If more than 49 attempts rate 'no' announcements, only the first 49 are kept.

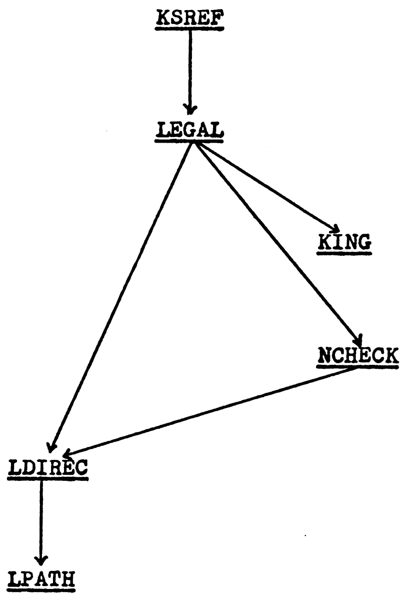

Fig. 2 A block flowchart
An arrow indicates a call of the routine at the head of the arrow by the one at the tail.

always executed by calling subroutine KSREF. Calls for the initialisation of processing and for try requests require the code in the first two sections of KSREF only. The remainder of KSREF and the other routines process attempted moves. KSREF calls function LEGAL to test legality of an attempt. For a 'no' or a 'heck-no', all needed changes to the output arrays MINFO and NOS are made by LEGAL. For a legal move, LEGAL clears and partially resets MINFO; KSREF finishes the task by determining mates and pawn tries for the next move. KSREF calls LEGAL to test possible pawn moves for captures. Finally, mating possibilities are tested. If a pawn capture exists, the game cannot be over. Otherwise, LEGAL tests moves generated by KSREF. If none are legal, there must be checkmate or stalemate (depending on whether the king is in check).

Functions LEGAL, LDIREC, and LPATH determine the legality of an attempted move. LEGAL is called with the attempt given in IDATA and with an argument distinguishing player-generated from program-generated moves. Less computation is required for the program-generated moves, which occur when testing pawn try and mating situations.

After testing the consistency of the input data, LEGAL calls function LDIREC, which determines if the piece is moving in a proper direction. If so, LDIREC calls LPATH to test the occupancy of squares along the path of the move. If the attempt passes both these tests, LEGAL uses NCHECK to make sure the king is not left in check. During castling attempts, NCHECK is also called to test program control of the king's passage square. For program-generated moves, LEGAL returns immediately after the status of the move has been deduced.

For player-generated moves, LEGAL must update the output arrays. If the move is illegal, LEGAL assigns a 'no' unless the attempt repeats an earlier one, represents an impossible attempted pawn try, or when the player does not attempt an obligatory pawn try under Eastern rules. These three cases are assigned a 'heck-no'. LEGAL records the determination in MINFO. For a legal move, MINFO is reset with the results of the move, NCHECK is called to find checks of the opponent's king, and IBOARD and ICOLOR are updated to reflect the move.

Function NCHECK determines whether a given square is controlled by a specified player. The numbers and directions of control are found. Because we use this function only for check situations, the number of possible controls to be found is limited by an input argument. In particular, Eastern rules never require more than one control to be found.

The function KING locates a specified king. NSIGN is a necessary number-theoretic function not found in standard FORTRAN libraries.

Use as a chess referee

These routines can, of course, be used to test the legality of chess moves. If the game is not started from its first move, the user must supply a complete two-level board (IBOARD), NOS(1) should be set to one, and ICOLOR set to indicate the player to move. Selection of rules is arbitrary, but execution is faster under Eastern rules. MTRYEA should be set to zero and ITRYRQ should always be zero. KSREF may now be called.

If the user wants no output other than an updated board and an indication of legality, he may by-pass KSREF and call LEGAL directly. The input should be as described above. If argument ITYPE is set to one, IBOARD, ICOLOR, and MINFO will be updated. If ITYPE equals two, only the function value will be returned. IBOARD and MINFO will be unchanged.

References

Wetherell, C. S., Buckholtz, T. J., and Booth, K. S. (1972). A director for *Kriegspiel*, a variant of chess, *The Computer Journal*, Vol. 15, No. 1, pp. 66-70.

Bell, A. G. (1970). Algorithm 50, How to program a computer to play legal chess, *The Computer Journal*, Vol. 13, No. 3, pp. 208-219.

Appendix: A driver program

This appendix presents a simple driver for the Kriegspiel referee. Inputs are the rules set in use and a sextuple specifying each move of the game. Outputs are an echo of the move, the board after the attempt is evaluated, the array MINFO, the color of the player to move next, and the reason for any game termination.

The program assumes two teletype-like input/output devices, with input on FORTRAN unit 1 and output on FORTRAN unit 2. These can, of course, be adjusted to the local environment. The program also assumes that an integer variable is capable of holding five characters of Hollerith data.

```
      CØMMØN /KSINP/ IDATA(8), IBØARD(8,8,2)
      CØMMØN /KSØUT/ MINFØ(30,2), NØS(50)
      EQUIVALENCE (IDATA(1),IRULES), (IDATA(2),ICØLØR)
C
C THIS PRØGRAM IS A DRIVER FØR THE KRIEGSPIEL REFEREE
C DESCRIBED IN THE PAPER 'A REFEREE FØR KRIEGSPIEL AND
C CHESS' BY BUCKHØLTZ AND WETHERELL.
C
C THE DRIVER ØPERATES BY READING MØVES FRØM INPUT UNIT IN
C AND PRINTING THE RESULTS ØN ØUTPUT UNIT ØUT.  BEFØRE A
C GAME IS BEGUN, THE USER IS ASKED TØ SPECIFY WESTERN ØR
C EASTERN RULES.  ØNCE THE GAME IS IN PRØGRESS, THE PLAYER
C TØ MØVE IS IDENTIFIED AND A MØVE IS REQUESTED.  AFTER THE
C ATTEMPTED MØVE IS EVALUATED, THE BØARD IS PRINTED, THE
C ARRAY MINFØ IS PRINTED AS A LIST ØF INTEGERS TØ BE
C INTERPRETED WITH THE AID ØF TABLE 3 ØF BUCKHØLTZ AND
C WETHERELL, AND IF THE MØVE ENDS THE GAME, THE REASØN FØR
C TERMINATIØN IS PRINTED.
C
C IN THE FØLLØWING DATA STATEMENTS, IT IS ASSUMED THAT
C INTEGER VARIABLES WILL HØLD 5 CHARACTERS ØF ALPHABETIC
C DATA.  THE INPUT UNIT IS GIVEN VALUE 1 AND THE ØUTPUT UNIT
C VALUE 2.
C
      DIMENSIØN NTEMP(8), NRULES(2), NCØLØR(2), NPIECE(13)
      INTEGER ØUT
      DATA ØUT/2/
      DATA NRULES/4HWEST, 4HEAST/
      DATA NCØLØR/5HWHITE, 5HBLACK/
      DATA NPIECE/2HBR, 2HBQ, 2HBB, 2HBN, 2HBP, 2HBK, 2H..,
     1            2HWK, 2HWP, 2HWN, 2HWB, 2HWQ, 2HWR/
      DATA IN/1/
C
C IN THIS FIRST SECTIØN, WE ASK FØR THE RULES SET TØ BE USED
C AND INITIALIZE THE GAME.  THE INITIALIZATIØN ØCCURS
C AS THE FIRST STATEMENT ØF THE LØØP REQUESTING MØVE
C SPECIFICATIØN.
C
```

```
    1 WRITE (OUT,2000)
      READ (IN,1000) IRULES
      IF (IRULES .NE. 1 .AND. IRULES .NE. 2) GOTO 1
      WRITE (OUT,2010) IRULES, NRULES(IRULES)
      ICOLOR = 0
C
C IN THIS SECTION, BOARD AND MINFO ARE PRINTED, TERMINATION
C TESTED, THE PLAYER TO MOVE ANNOUNCED, AND INPUT REQUESTED.
C KSREF IS CALLED TO ARBITRATE THE MOVE.
C
    2 CALL KSREF
      DO 4 I = 1,8
        L = 9 - I
        DO 3 J = 1,8
          K = IBOARD(J,L,1)
    3     NTEMP(J) = NPIECE(K+7)
    4   WRITE (OUT,2020) NTEMP
      WRITE (OUT,2030)
      WRITE (OUT,2040) (I,I=1,30), (J,(MINFO(I,J),I=1,30),
     1                                                  J=1,2)
      IF (MINFO(1,1) .NE. 1) GOTO 5
      I = 1
      IF (ICOLOR .EQ. -1) I = 2
      WRITE (OUT,2050) NCOLOR(I)
      READ (IN,1010) (IDATA(I),I=3,8)
      WRITE (OUT,2060) IDATA
      GOTO 2
C
C THIS SECTION ENDS THE GAME.
C
    5 WRITE (OUT,2070) MINFO(1,1)
      STOP
C
C FORMAT STATEMENTS FOR INPUT.
C
 1000 FORMAT(I1)
 1010 FORMAT(6I1)
C
C FORMAT STATEMENTS FOR OUTPUT.
C
 2000 FORMAT(36H1 PLEASE TYPE 1 FOR WESTERN OR 2 FOR
     1        15H EASTERN RULES.)
 2010 FORMAT(11H INPUT WAS  I1,  5H FOR  A4, 10HERN RULES.)
 2020 FORMAT(1X,8(A2, 1X))
 2030 FORMAT(19H THE ARRAY MINFO IS)
 2040 FORMAT(6X,30I3/ (2X, I1, 1X, 30I3))
 2050 FORMAT(1X, A5,  8H TO MOVE/23H PLEASE TYPE YOUR MOVE.)
 2060 FORMAT(30H THE COMPLETE VECTOR IDATA IS 8I2)
 2070 FORMAT(25H THE TERMINATION CODE IS ,I1)
      END


      SUBROUTINE KSREF
      COMMON /KSINP/ IDATA(8),IBOARD(8,8,2)
      COMMON /KSOUT/ MINFO(30,2),NOS(50)
      EQUIVALENCE (IDATA(1),IRULES),(IDATA(2),ICOLOR),
     1            (IDATA(3),IFROMH),(IDATA(4),IFROMV),
     2            (IDATA(5),ITOH),(IDATA(6),ITOV),
     3            (IDATA(7),IPROMO),(IDATA(8),ITRYRQ)
      EQUIVALENCE (MINFO(1,1),MTERMN),(MINFO(1,2),MRULES),
     1            (MINFO(2,1),MLOST),(MINFO(3,1),MSQRH),
     2            (MINFO(3,2),MSQRV),(MINFO(4,1),MCAPT),
     3            (MINFO(5,1),MPROMO),(MINFO(6,1),MATMPT),
     4            (MINFO(6,2),MCOLOR),(MINFO(7,1),MCHEKA),
     5            (MINFO(7,2),MCHEKB),(MINFO(8,1),MTRYEA),
     6            (MINFO(8,2),MNOS),(MINFO(9,1),MSTRY),
     7            (MINFO(24,1),MDTRY)
C
C
C THIS SUBROUTINE ACTS AS A REFEREE FOR THE GAME OF
C KRIEGSPIEL, A VARIANT OF CHESS.
C
C THE SUBROUTINE IS CALLED TO START GAME PROCESSING, TO
C PROCESS PAWN TRY REQUESTS, AND TO TEST THE LEGALITY OF
C ATTEMPTED MOVES.
C
C THE INPUT AND OUTPUT ITEMS ARE PASSED THROUGH COMMON
C BLOCKS KSINP AND KSOUT.  EACH COMMON BLOCK CONTAINS BOTH
C INPUT AND OUTPUT ITEMS.
C
C IN GENERAL, THE CALLING PROGRAM PROVIDES AN ATTEMPTED MOVE
C IN VECTOR IDATA.  KSREF USES INFORMATION MAINTAINED IN THE
C REST OF THE COMMON BLOCKS TO PROCESS THE ATTEMPT.  AN
C ATTEMPT WILL CAUSE A CHANGE IN THE OUTPUT DATA ITEMS WHICH
C REPORT THE RESULTS OF THE ATTEMPT.  THE CALLING PROGRAM
C SHOULD NOT DESTROY THE CONTENTS OF IBOARD, MINFO, OR NOS
C DURING A GAME, OR ATTEMPTED MOVES WHICH DEPEND ON HISTORY
C MAY NOT BE HANDLED PROPERLY.
C
C THE ARRAY IDATA SELECTS THE TYPE OF PROCESSING AND GIVES
C BASIC CONTROL PARAMETERS.  THE ELEMENTS ARE
C    IRULES = RULES SET IN USE
C    ICOLOR = COLOR OF PLAYER TO MOVE
C    (IFROMH,IFROMV) = INITIAL SQUARE OF MOVE
C    (ITOH,ITOV) = DESTINATION SQUARE OF MOVE
C    IPROMO = PIECE SPECIFIER FOR PAWN PROMOTION
C    ITRYRQ = FLAG FOR PAWN TRY REQUEST PROCESSING.
C VARIABLE IRULES IS NEVER CHANGED BY KSREF.  VARIABLE
C ICOLOR IS NEGATED WHEN A LEGAL MOVE IS MADE AND SET TO
C ZERO WHEN A GAME ENDING MOVE IS MADE.
C
C THE ARRAY IBOARD CONTAINS THE REFEREE'S CHESS BOARD.
C IBOARD(I,J,1) GIVES THE PIECE ON SQUARE (I,J) AND
C IBOARD(I,J,2) RECORDS NECESSARY INFORMATION ABOUT THE
C PIECE.

C
C THE ARRAY MINFO PROVIDES DATA TO BE DISPLAYED TO THE
C PLAYERS.  SOME ITEMS ARE USED AS INPUT TO LATER CALLS OF
C KSREF.  THE ELEMENTS ARE DESCRIBED IN TABLE 3 OF THE
C ACCOMPANYING ARTICLE.
C
C THE ARRAY NOS COLLECTS KRIEGSPIEL 'NO' ATTEMPTS.  THESE
C ARE USED TO FIND FUTURE 'HECK NO' MOVES.  THE ARRAY IS
C CLEARED AFTER A LEGAL ATTEMPT.
C
C
C THE VARIABLES ICOLOR AND ITRYRQ DETERMINE THE PROCESSING
C REQUESTED OF KSREF.  THE FOLLOWING IF STATEMENTS
C DIFFERENTIATE THE CASES.  FALL-THROUGH CAUSES GAME
C INITIALIZATION.
      IF (ICOLOR .NE. 0 .AND. ITRYRQ .EQ. 1) GOTO 5
      IF (ICOLOR .NE. 0 .AND. ITRYRQ .EQ. 0) GOTO 6
C
C THIS SECTION PERFORMS INITIALIZATION OF THE ARRAYS IBOARD,
C MINFO, AND NOS.  THE MEN ARE PLACED IN STARTING POSITIONS,
C THE HISTORY CELLS OF IBOARD ARE SET TO ZERO, AND THE FIRST
C DATA DISPLAY IS GENERATED.  ICOLOR IS SET TO 1 INDICATING
C WHITE TO MOVE.  NUMERICAL CODES FOR THE PIECES ARE GIVEN
C IN FIGURE 1.
      IBOARD(8,1,1) = 6
      IBOARD(1,1,1) = 6
      IBOARD(2,1,1) = 3
      IBOARD(7,1,1) = 3
      IBOARD(3,1,1) = 4
      IBOARD(6,1,1) = 4
      IBOARD(4,1,1) = 5
      IBOARD(5,1,1) = 1
      DO 1 I = 1,8
        IBOARD(I,2,1) = 2
        IBOARD(I,7,1) = -2
    1   IBOARD(I,8,1) = -IBOARD(I,1,1)
      DO 3 I = 1,8
        DO 2 J = 3,6
    2     IBOARD(I,J,1) = 0
        DO 3 J = 1,8
    3     IBOARD(I,J,2) = 0
      DO 4 J = 1,2
        DO 4 I = 1,30
    4     MINFO(I,J) = 0
      MTERMN = 1
      MCOLOR = 1
      MRULES = IRULES
      NOS(1) = 1
      ICOLOR = 1
      RETURN
C
C THIS SECTION PROCESSES PAWN TRY REQUESTS, WHICH MAY OCCUR
C ONLY UNDER EASTERN RULES.  MTRYEA RETURNS THE VALUE 2 IF
C IF A PAWN CAPTURE EXISTS AND 1 OTHERWISE.  FOR EASTERN
C RULES, MSTRY = MINIMUM(1,NUMBER OF EXISTING PAWN CAPTURES).
    5 MTRYEA = MSTRY + 1
      RETURN
C
C THE REST OF KSREF PROCESSES ATTEMPTED MOVES.  FUNCTION
C LEGAL TESTS THE LEGALITY OF THE MOVE GIVEN IN IDATA.  IF
C THE MOVE IS ILLEGAL, ARRAY MINFO REFLECTS THE CAUSE AND
C KSREF RETURNS IMMEDIATELY.
C
C IF THE MOVE IS LEGAL, ARRAYS IBOARD, MINFO, AND NOS
C CONTAIN THE UPDATED GAME STATUS.  IBOARD AND MINFO MUST BE
C COMPLETED BY RECORDING PAWN TRIES RELEVANT TO THE NEXT
C MOVE.  FUNCTION LEGAL REVERSES ICOLOR IF THE MOVE IS
C LEGAL.
    6 IF (LEGAL(1) .NE. 1) RETURN
C
C THIS SECTION GENERATES PAWN CAPTURES AVAILABLE ON THE NEXT
C MOVE AND ENTERS THEM IN MINFO.  FIRST A SEARCH IS MADE FOR
C PAWNS OF PLAYER ICOLOR.  WHEN A PAWN IS FOUND, ITS TWO
C POSSIBLE CAPTURE MOVES ARE TESTED USING FUNCTION LEGAL.
C IF A LEGAL CAPTURE IS FOUND, MSTRY IS UPDATED AND THE
C COORDINATES ARE RECORDED IN THE SINGLE TRY SECTION OF
C MINFO.  AFTER THE FIRST CAPTURE, ADDITIONAL CAPTURES CAUSE
C A TEST FOR POSSIBLE DOUBLE TRIES.  NOTE THAT IN EASTERN
C RULES, THE LOOP CAN BE LEFT ONCE ANY CAPTURE IS NOTED IN
C MSTRY.
      DO 11 IFROMV = 2,7
        ITOV = IFROMV + ICOLOR
        DO 11 IFROMH = 1,8
          IF (ICOLOR*IBOARD(IFROMH,IFROMV,1) .NE. 2) GOTO 11
          DO 10 I = 1,3,2
            ITOH = IFROMH + 2 - I
            IF (LEGAL(2) .NE. 1) GOTO 10
            IF (MSTRY .EQ. 0) GOTO 9
            DO 8 J = 1,MSTRY
              IF (MINFO(J+9,1) .NE. ITOH .OR.
     1            MINFO(J+9,2) .NE. ITOV) GOTO 8
              MDTRY = MDTRY + 1
            MINFO(MDTRY+24,1) = ITOH
            MINFO(MDTRY+24,2) = ITOV
            DO 7 K = J,MSTRY
              MINFO(K+9,1) = MINFO(K+10,1)
  7           MINFO(K+9,2) = MINFO(K+10,2)
            MSTRY = MSTRY - 1
            GOTO 10
  8         CONTINUE
  9       MSTRY = MSTRY + 1
          IF (IRULES .EQ. 2) GOTO 12
          MINFO(MSTRY+9,1) = ITOH
          MINFO(MSTRY+9,2) = ITOV
 10       CONTINUE
 11     CONTINUE

C
C FINALLY, THIS SECTION TESTS FOR CHECKMATE AND STALEMATE.
C IF THERE IS A PAWN TRY AVAILABLE, THERE IS NO MATE.  IF
C THERE ARE NO PAWN TRIES, THE CODE IN THE DO LOOP FINDS
C EACH OF PLAYER ICOLOR'S MEN AND MOVES IT TO EVERY
C POSSIBLE SQUARE OF THE BOARD.  IF NO LEGAL MOVE IS FOUND,
C THE KING IS MATED.  MCHEKA DISTINGUISHES CHECKMATE FROM
C STALEMATE.  MTERMN RETURNS A CODE INDICATING CONTINUATION
C OR TYPE OF TERMINATION OF THE GAME.  ICOLOR IS SET TO ZERO
C AT TERMINATION.
   12 MTERMN = 1
      IF (MSTRY + MDTRY .GT. 0) RETURN
      DO 14 IFROMH = 1,8
        DO 14 IFROMV = 1,8
          IF (IBOARD(IFROMH,IFROMV,1)*ICOLOR .LE. 0) GOTO 14
          DO 13 ITOH = 1,8
            DO 13 ITOV = 1,8
              IF (LEGAL(2) .GT. 0) RETURN
 13           CONTINUE
 14     CONTINUE
      MTERMN = 2
      IF (MCHEKA .EQ. 0) MTERMN = 3
      ICOLOR = 0
      RETURN
      END
C
      FUNCTION LEGAL(ITYPE)
      COMMON /KSINP/ IDATA(8),IBOARD(8,8,2)
      COMMON /KSOUT/ MINFO(30,2),NOS(50)
      EQUIVALENCE (IDATA(1),IRULES),(IDATA(2),ICOLOR),
     1            (IDATA(3),IFROMH),(IDATA(4),IFROMV),
     2            (IDATA(5),ITOH),(IDATA(6),ITOV),
     3            (IDATA(7),IPROMO),(IDATA(8),ITRYRQ)
      EQUIVALENCE (MINFO(1,1),MTERMN),(MINFO(1,2),MRULES),
     1            (MINFO(2,1),MLOST),(MINFO(3,1),MSQRH),
     2            (MINFO(3,2),MSQRV),(MINFO(4,1),MCAPT),
     3            (MINFO(5,1),MPROMO),(MINFO(6,1),MATMPT),
     4            (MINFO(6,2),MCOLOR),(MINFO(7,1),MCHEKA),
     5            (MINFO(7,2),MCHEKB),(MINFO(8,1),MTRYEA),
     6            (MINFO(8,2),MNOS),(MINFO(9,1),MSTRY),
     7            (MINFO(24,1),MDTRY)
C
C
C THIS FUNCTION TESTS THE LEGALITY OF AN ATTEMPT.  IF THE
C MOVE IS LEGAL, THE RETURN VALUE IS 1, IF A NO, THE VALUE
C IS 0, AND IF A HECK-NO, THE VALUE IS -1.
C
C LEGAL TESTS THE VALIDITY OF BOTH PLAYER- AND PROGRAM-
C GENERATED ATTEMPTS.  THE INPUT ARGUMENT ITYPE IS 1 FOR A
C PLAYER MOVE AND 2 FOR A PROGRAM MOVE.  FOR ITYPE = 1 AND
C A LEGAL MOVE, THE MOVE IS MADE ON IBOARD AND THE UPDATING
C OF IBOARD, NOS, AND MINFO BEGINS.  FOR ITYPE = 1 AND AN
C ILLEGAL ATTEMPT, VALUES OF MINFO AND NOS ARE SET TO


C INDICATE THE TYPE AND CAUSE OF ERROR.  IF ITYPE = 2, THE
C FUNCTION VALUE IS THE ONLY OUTPUT.
C
C THE ATTEMPTED MOVE IS FOUND IN IDATA ENTRIES
C (IFROMH,IFROMV), (ITOH,ITOV), AND IPROMO.
C
C IF ITYPE = 1 AND THE MOVE IS SUCCESSFUL, THE SIGN OF
C ICOLOR IS REVERSED.  OTHERWISE, IDATA IS UNCHANGED UPON
C RETURN.
C
C
C THE FIRST TEST REQUIRES THAT BOTH INITIAL AND DESTINATION
C SQUARES BE ON THE BOARD, THAT A FRIENDLY MAN BE IN THE
C INITIAL SQUARE, THAT THE MAN ACTUALLY MOVE, AND THAT THE
C DESTINATION SQUARE NOT CONTAIN A FRIENDLY MAN.
      JMAN = IABS(IBOARD(IFROMH,IFROMV,1))
      JH = ITOH - IFROMH
      JV = ITOV - IFROMV
      IF (IFROMH .LT. 1 .OR. IFROMH .GT. 8) GOTO 27
      IF (IFROMV .LT. 1 .OR. IFROMV .GT. 8) GOTO 27
      IF (ITOH .LT. 1 .OR. ITOH .GT. 8) GOTO 27
      IF (ITOV .LT. 1 .OR. ITOV .GT. 8) GOTO 27
      IF (ICOLOR*IBOARD(IFROMH,IFROMV,1) .LE. 0) GOTO 27
      IF (JH .EQ. 0 .AND. JV .EQ. 0) GOTO 27
      IF (ICOLOR*IBOARD(ITOH,ITOV,1) .GT. 0) GOTO 27
C
C THIS SECTION TESTS CONSISTENCY OF PROMOTION REQUESTS.  IF
C THE MOVE IS PROGRAM GENERATED, PROMOTIONS ARE IGNORED.
C PAWN MOVES TO THE EIGHTH RANK MUST SPECIFY PROMOTION AND
C ALL OTHERS MUST HAVE A ZERO VALUE IN IPROMO.
      IF (ITYPE .EQ. 2) GOTO 5
      IF (IPROMO .EQ. 0) GOTO 1
      IF (JMAN .NE. 2) GOTO 27
      IF (ITOV .NE. 1 .AND. ITOV .NE. 8) GOTO 27
      IF (IPROMO .LT. 3 .OR. IPROMO .GT. 6) GOTO 27
      GOTO 2
    1 IF (JMAN .EQ. 2 .AND. (ITOV .EQ. 1 .OR. ITOV .EQ. 8))
     1  GOTO 27
C
C UNDER EASTERN RULES, ONLY A PAWN CAPTURE MAY BE ATTEMPTED
C AFTER AN OF SOME ANSWER TO A TRY REQUEST.  IF A NONE
C ANSWER WAS RETURNED, A PAWN CAPTURE ATTEMPT IS A HECK-NO.
C THISSECTION FIRST TESTS FOR EASTERN RULES.  THE VARIABLE
C J IS SET TO ZERO IF AND ONLY IF A PAWN IS MOVING ONE
C SQUARESIDEWAYS.  (VERTICAL MOTION IS TESTED LATER.)  A
C THREE-WAY BRANCH DECIDES WHETHER TRIES HAVE BEEN REQUESTED
C AND DETERMINES THE RESULTS OF THE REQUEST.  THE LAST TWO
C STATEMENTS USE J TO DECIDE LEGALITY.
    2 IF (IRULES .EQ. 1) GOTO 5
      J = IABS(JMAN-2) + IABS(IABS(JH)-1)
      IF (MTRYEA-1) 5,3,4
    3 IF (J) 5,27,5
    4 IF (J) 27,5,27


C
C THE FUNCTION LDIREC RETURNS A NEGATIVE VALUE IF THE
C ATTEMPTED MOVE IS A HECK-NO BY VIRTUE OF A VIOLATION OF
C PIECE MOTION RULES, A ZERO VALUE IF THE MOVE IS A SIMPLE
C NO, AND A POSITIVE VALUE FOR LEGAL MOVES.  LDIREC DOES NOT
C TEST FOR A KING MOVING INTO CHECK OR (FOR CASTLING)
C THROUGH CHECK.  JNPASS IS RETURNED WITH VALUE 2 IF AN
C APPARENTLY LEGAL EN PASSANT PAWN TRY WAS ATTEMPTED AND
C WITH VALUE 1 OTHERWISE.
    5 IF (LDIREC(JMAN,JNPASS)) 27,23,6
C
C THIS SECTION DETERMINES IF THE KING HAS BEEN LEFT IN CHECK
C BY THE ATTEMPTED MOVE OR IF, IN THE CASE OF CASTLING, THE
C PASSAGE SQUARE IS CONTROLLED BY THE OPPONENT.  THE MOVE IS
C MADE PROVISIONALLY, THE KING LOCATED, AND THE NECESSARY
C SQUARES TESTED.  SEVERAL BOARD LOCATIONS ARE SAVED.  ONLY
C A PLAYER-GENERATED LEGAL MOVE CAUSES A RETURN WITH AN
C UPDATED BOARD.  THE MOVE IS MADE AND THE KING IS LOCATED.
C THE KING'S SQUARE (AND IF HE CASTLED, HIS PASSAGE SQUARE)
C IS TESTED BY NCHECK.  IF THESE TESTS ARE PASSED, THE MOVE
C IS LEGAL.
    6 JTHFV1 = IBOARD(ITOH,IFROMV,1)
      JTHFV2 = IBOARD(ITOH,IFROMV,2)
      JTHTV1 = IBOARD(ITOH,ITOV,1)
      JFHFV1 = IBOARD(IFROMH,IFROMV,1)
      IF (JNPASS .NE. 2) GOTO 7
      IBOARD(ITOH,IFROMV,1) = 0
      IBOARD(ITOH,IFROMV,2) = 0
    7 IBOARD(ITOH,ITOV,1) = IBOARD(IFROMH,IFROMV,1)
      IBOARD(IFROMH,IFROMV,1) = 0
      LEGAL = 1
      CALL KING(JKH,JKV)
      IF (NCHECK(JKH,JKV,1,JDUM,JDUM) .NE. 0) GOTO 8
      IF (JMAN .NE. 1 .OR. IABS(JH) .NE. 2) GOTO 9
      IF (NCHECK(JKH-JH/2,ITOV,1,JDUM,JDUM)
     1    .EQ. 0) GOTO 9
C
C THIS SECTION RESETS IBOARD WHEN THE KING IS LEFT IN CHECK
C OR CASTLES THROUGH CHECK.  CONTROL THEN GOES TO THE NO
C MOVE PROCESSOR.
    8 IBOARD(ITOH,IFROMV,1) = JTHFV1
      IBOARD(ITOH,IFROMV,2) = JTHFV2
      IBOARD(ITOH,ITOV,1) = JTHTV1
      IBOARD(IFROMH,IFROMV,1) = JFHFV1
      GOTO 23
C
C IF THE MOVE WAS LEGAL BUT PROGRAM-GENERATED, THE BOARD
C MUST BE RESET AND LEGAL RETURNS WITH VALUE 1 FROM ABOVE.
    9 IF (ITYPE .EQ. 1) GOTO 10
      IBOARD(ITOH,IFROMV,1) = JTHFV1
      IBOARD(ITOH,IFROMV,2) = JTHFV2
      IBOARD(ITOH,ITOV,1) = JTHTV1
      IBOARD(IFROMH,IFROMV,1) = JFHFV1


      RETURN
C
C THIS SECTION, REACHED ONLY FOR PLAYER-GENERATED LEGAL
C MOVES, CONTINUES TO UPDATE THE BOARD AS BEGUN WHEN TESTING
C CHECK ABOVE.  HISTORY SQUARES ARE UPDATED AS EXPLAINED IN
C TABLE 2.
   10 IBOARD(IFROMH,IFROMV,2) = 0
      GOTO (11,13,16,16,16,15), JMAN
   11 IF (IABS(JH) .NE. 2) GOTO 15
      JDIREC = JH/2
      JCASTL = IFROMH + JDIREC
      IBOARD(JCASTL,IFROMV,1) = 6*ICOLOR
      IBOARD(JCASTL,IFROMV,2) = 1
      ICORNR = MIN0(8,JCASTL+3*JDIREC)
      IBOARD(ICORNR,IFROMV,1) = 0
   12 IBOARD(ITOH,ITOV,2) = 1
      GOTO 18
   13 IF (IABS(JV) .EQ. 2) GOTO 15
      IF (IPROMO .NE. 0) GOTO 14
      IBOARD(ITOH,ITOV,2) = 2
      GOTO 17
   14 IBOARD(ITOH,ITOV,1) = ICOLOR*IPROMO
      IF (IPROMO .NE. 6) GOTO 16
   15 IBOARD(ITOH,ITOV,2) = 1
      GOTO 17
   16 IBOARD(ITOH,ITOV,2) = 0
```

```
C
C THE VARIABLES JKH AND JKV RETAIN THE LOCATION OF THE KING.
   17 IBOARD(JKH,JKV,2) = MOD(IBOARD(JKH,JKV,2),2)
   18 JPAWNV = 4 + (ICOLOR+1)/2
      DO 19 JPAWNH = 1,8
   19   IF (IABS(IBOARD(JPAWNH,JPAWNV,1)) .EQ. 2)
     1      IBOARD(JPAWNH,JPAWNV,2) = 2
C
C THIS SECTION CLEARS THE ARRAY NOS, REVERSES OUTPUT
C VARIABLE ICOLOR, AND FILLS ARRAY MINFO WITH DATA TO BE
C DISPLAYED.  THE VARIABLE JTHTV1 IDENTIFIES ANY
C CAPTURED MAN.
      NOS(1) = 1
      ICOLOR = -ICOLOR
      DO 20 K = 1,2
        DO 20 J = 1,30
   20     MINFO(J,K) = 0
      CALL KING(JKH,JKV)
      IBOARD(JKH,JKV,2) = IBOARD(JKH,JKV,2)
     1  + 2*NCHECK(JKH,JKV,3-IRULES,MCHEKA,MCHEKB)
      MRULES = IRULES
      MCOLOR = 2 - (ICOLOR+1)/2
      IF (IPROMO .NE. 0) MPROMO = 1
      IF (JNPASS .EQ. 1) GOTO 21
      MLOST = 2
      JCAPTV = ITOV + ICOLOR
      GOTO 22


   21 IF (JTHTV1 .EQ. 0) RETURN
      MLOST = IABS(JTHTV1)
      JCAPTV = ITOV
   22 MSQRH = ITOH
      MSQRV = JCAPTV
      MCAPT = 2
      IF (MLOST .NE. 2) MCAPT = 1
      IF (IRULES .EQ. 2) MCAPT = 3
      RETURN
C
C THIS SECTION PROCESSES SIMPLE NO MOVES.  LEGAL IS SET TO
C ZERO.  IF ITYPE FLAGS A PROGRAM-GENERATED MOVE AN
C IMMEDIATE RETURN IS TAKEN.
C
C A NO IS A HECK-NO IF AN UNANNOUNCED PAWN CAPTURE IS TRIED
C UNDER WESTERN RULES.  THE CODE TO STATEMENT 25 TESTS THIS
C POSSIBILITY.
C
C IN EITHER RULES SET, A NO IS A HECK-NO IF IT WAS REJECTED
C ON AN EARLIER ATTEMPT.  ARRAY NOS CONTAINS CODED DATA
C CONCERNING THE NOS SINCE THE LAST LEGAL MOVE.  THE CODING
C METHOD IS EXPLAINED IN TABLE 4.
C
C IF THE NO SURVIVES THESE TESTS, IT IS ENTERED INTO ARRAY
C NOS.  MINFO RECORDS THE REJECTION.  ONLY 50 ITEMS
C INCLUDING COUNT ARE ALLOWED IN NOS.
C
   23 LEGAL = 0
      IF (ITYPE .EQ. 2) RETURN
      IF (IRULES .EQ. 2) GOTO 25
      IF (JMAN .NE. 2) GOTO 25
      IF (IABS(JH) .NE. 1) GOTO 25
      IF (MSTRY .EQ. 0) GOTO 27
      DO 24 I = 1,MSTRY
        IF (ITOH .EQ. MINFO(I+9,1)
     1      .AND. ITOV .EQ. MINFO(I+9,2)) GOTO 25
   24   CONTINUE
      GOTO 27
   25 NOCODE = 729*IFROMH + 81*IFROMV + 9*ITOH + ITOV
      J = NOS(1)
      DO 26 I = 1,J
        IF (NOCODE .EQ. NOS(I)) GOTO 27
   26   CONTINUE
      MNOS = MNOS + 1
      MATMPT = 1
      IF (J .GE. 50) RETURN
      NOS(1) = J + 1
      NOS(J+1) = NOCODE
      RETURN
C
C FINALLY, THIS SECTION HANDLES HECK-NO'S.  LEGAL IS SET TO
C -1 AND FOR PLAYER-GENERATED MOVES, MATMPT = 2.
   27 LEGAL = -1


      IF (ITYPE .EQ. 1) MATMPT = 2
      RETURN
      END
C
      FUNCTION LDIREC(JMAN,JNPASS)
      COMMON /KSINP/ IDATA(8),IBOARD(8,8,2)
      EQUIVALENCE (IDATA(1),IRULES),(IDATA(2),ICOLOR),
     1            (IDATA(3),IFROMH),(IDATA(4),IFROMV),
     2            (IDATA(5),ITOH),(IDATA(6),ITOV),
     3            (IDATA(7),IPROMO),(IDATA(8),ITRYRQ)
C
C FUNCTION LDIREC TESTS FOR LEGAL MOVES UNDER THE PIECE
C MOTION LAWS.  THE VALUE IS 1 FOR LEGAL MOVES, 0 FOR SIMPLE
C NO MOVES, AND -1 FOR HECK-NO MOVES.  INPUT ARGUMENT JMAN
C GIVES THE PIECE MAKING THE MOVE.  ARGUMENT JNPASS
C RETURNS A VALUE OF 2 FOR A LEGAL EN PASSANT PAWN TRY AND A
C VALUE OF 1 OTHERWISE.
C
```

```
C LDIREC MAKES USE OF FUNCTION LPATH TO SEE IF THE PATH
C TAKEN BY THE PIECE IS CLEAR.  THE VALUE IS 4 IF THE WHOLE
C PATH IS CLEAR, 3 IF THE FINAL SQUARE CONTAINS AN
C OPPONENT'S MAN, 2 IF AN INTERMEDIATE SQUARE CONTAINS AN
C OPPONENT'S MAN, AND 1 IF ANY SQUARE CONTAINS A FRIENDLY
C MAN.  IF TWO VALUES ARE POSSIBLE, THE LESSER IS TAKEN.
C
C AFTER CALCULATING THE DISPLACEMENT OF THE PIECE TO MOVE,
C THE ROUTINE TRANSFERS TO A SUBSECTION WHICH HANDLES THE
C PIECE JMAN.  EACH OF THESE SECTIONS TESTS FOR A LEGAL
C DIRECTION OF MOVE AND THEN CALLS LPATH TO TEST FOR A CLEAR
C PATH.  THE PAWN SECTION HAS SOME ADDITIONAL COMPLEXITIES.
C
C LDIREC DOES NOT TEST FOR OPPONENT'S CONTROL OF ANY SQUARE.
C TESTS FOR MOVING INTO OR THROUGH CHECK ARE MADE, WHEN
C NECESSARY, BY LEGAL.
C
      IDLTAH = ITOH - IFROMH
      IDLTAV = ITOV - IFROMV
      JNPASS = 1
      GOTO (1,3,9,10,12,13), JMAN
C
C IN KING MOVES, THE FIRST TEST IS FOR TENTATIVE CASTLING.
C NEITHER STEP MAY BE GREATER THAN 1 EXCEPT FOR CASTLING.
C IN CASTLING, THE KING AND ROOK MUST HAVE HISTORIES OF
C ZERO.  THE KING MAY NOT MOVE VERTICALLY.  THE PATH
C BETWEEN KING AND ROOK MUST BE CLEAR.
    1 IF (IABS(IDLTAH) .EQ. 2) GOTO 2
      IF (IABS(IDLTAH) .GT. 1) GOTO 16
      IF (IABS(IDLTAV) .GT. 1) GOTO 16
      JPATH = LPATH(IDLTAH,IDLTAV,1)
      GOTO (16,14,14,14), JPATH
    2 IF (IDLTAV .NE. 0) GOTO 16
      IF (IBOARD(IFROMH,IFROMV,2) .NE. 0) GOTO 16
      ICORNR = MIN0(8,IFROMH+2*IDLTAH)



      IF (IBOARD(ICORNR,IFROMV,1) .NE. 6*ICOLOR) GOTO 16
      IF (IBOARD(ICORNR,IFROMV,2) .NE. 0) GOTO 16
      JPATH = LPATH(IDLTAH/2,0,IABS(ICORNR-IFROMH)-1)
      GOTO (16,15,15,14), JPATH
C
C IN PAWN MOVES, THE MAN MUST ALWAYS MOVE FORWARD AND MAY
C ONLY MOVE ONE OR TWO SQUARES.  TWO STEP MOVES MUST HAVE A
C CLEAR PATH AND NO HORIZONTAL COMPONENT.  NORMAL ONE-STEP
C MOVES MUST HAVE A CLEAR DESTINATION.  PAWNS ATTEMPTING
C CAPTURE MUST MOVE ONE STEP SIDEWAYS ALSO AND THERE MUST
C EITHER BE AN OPPONENT ON THE FINAL SQUARE OR AN OPPONENT'S
C PAWN AVAILABLE FOR EN PASSANT CAPTURE.
    3 IF (NSIGN(IDLTAV) .NE. ICOLOR) GOTO 16
      IF (IABS(IDLTAV) - 2) 5,4,16
    4 IF (IDLTAH .NE. 0) GOTO 16
      IF (IBOARD(IFROMH,IFROMV,2) .NE. 0) GOTO 16
      JPATH = LPATH(0,NSIGN(IDLTAV),2)
      GO TO (16,15,15,14), JPATH
    5 IF (IABS(IDLTAH) - 1) 6,7,16
    6 JPATH = LPATH(0,IDLTAV,1)
      GOTO (16,15,15,14), JPATH
    7 JPATH = LPATH(IDLTAH,IDLTAV,1)
      GOTO (16,14,14,8), JPATH
    8 IF (ICOLOR*IBOARD(ITOH,IFROMV,1) .GT. 0)
     1  GOTO (16,15), IRULES
      IF (IABS(IBOARD(ITOH,IFROMV,1)) .NE. 2 .OR.
     1   IBOARD(ITOH,IFROMV,2) .NE. 1) GOTO (16,15), IRULES
      JNPASS = 2
      GOTO 14
C
C THE PRODUCT OF THE DISPLACEMENTS OF A KNIGHT'S MOVE MUST
C ALWAYS HAVE ABSOLUTE VALUE 2.  LEGAL ALWAYS TESTS THAT
C THE DESTINATION IS ON THE BOARD.
    9 IF (IABS(IDLTAH*IDLTAV) .NE. 2) GO TO 16
      JPATH = LPATH(IDLTAH,IDLTAV,1)
      GOTO (16,14,14,14), JPATH
C
C A BISHOP'S MOVE MUST ALWAYS HAVE THE SAME ABSOLUTE
C DISPLACEMENT IN BOTH DIRECTIONS.
   10 IF (IABS(IDLTAH) .NE. IABS(IDLTAV)) GOTO 16
   11 JPATH = LPATH(NSIGN(IDLTAH),NSIGN(IDLTAV),
     1  MAX0(IABS(IDLTAH),IABS(IDLTAV)))
      GOTO (16,15,14,14), JPATH
C
C A QUEEN'S MOVE IS EITHER A ROOK'S MOVE OR A BISHOP'S MOVE.
   12 IF (IABS(IDLTAH) .EQ. IABS(IDLTAV)) GOTO 11
C
C A ROOK'S MOVE MUST HAVE ZERO AS THE PRODUCT OF THE
C DISPLACEMENT'S.
   13 IF (IDLTAH*IDLTAV) 16,11,16
C
C THE FOLLOWING LINES RETURN THE THREE POSSIBLE VALUES.
   14 LDIREC = 1



      RETURN
   15 LDIREC = 0
      RETURN
   16 LDIREC = -1
      RETURN
      END
C
      FUNCTION LPATH(IDLTAH,IDLTAV,ISTEPS)
      COMMON /KSINP/ IDATA(8),IBOARD(8,8,2)
      EQUIVALENCE (IDATA(2),ICOLOR),(IDATA(3),IFROMH),
     1            (IDATA(4),IFROMV)
C
C FUNCTION LPATH TESTS FOR THE PRESENCE OF PIECES ALONG THE
C PATH OF A PROJECTED MOVE.  THE PATH IS DEFINED BY THE
C INITIAL SQUARE (IFROMH,IFROMV), THE HORIZONTAL STEP IDLTAH
C AND THE VERTICAL STEP IDLTAV, AND THE NUMBER OF STEPS
C ISTEPS.  THE VALUE IS 4 IF THE WHOLE PATH IS CLEAR, 3 IF
C THE FINAL SQUARE CONTAINS AN OPPONENT'S MAN, 2 IF AN
C INTERMEDIATE SQUARE CONTAINS AN OPPONENT'S MAN, AND 1 IF
C ANY SQUARE CONTAINS A FRIENDLY MAN.  IF TWO VALUES ARE
C POSSIBLE, THE LOWER IS TAKEN.  (IFROMH,IFROMV) IS NOT PART
C OF THE PATH.  ISTEPS IS ALWAYS POSITIVE.
C
C THE BASIC METHOD IS TO STEP ALONG THE PATH, LOOKING FOR
C PIECES.  THE PRODUCT IBOARD(IHORIZ,IVRTCL,1)*ICOLOR IS
C NEGATIVE IF AN OPPONENT IS ON (IHORIZ,IVRTCL), ZERO IF THE
C SQUARE IS EMPTY, AND POSITIVE IF IT IS OCCUPIED BY A
C FRIENDLY MAN.  NUMOPP AND LPLACE ARE USED TO DISTINGUISH
C THE VARIOUS CASES FOR VALUES OF 2, 3, AND 4.
C
      NUMOPP = 0
      LPATH = 1
      DO 2 I = 1,ISTEPS
        IHORIZ = IFROMH + I*IDLTAH
        IVRTCL = IFROMV + I*IDLTAV
        IF (ICOLOR*IBOARD(IHORIZ,IVRTCL,1)) 1,2,5
    1   NUMOPP = NUMOPP + 1
        LPLACE = I
    2   CONTINUE
      LPATH = 4
      IF (NUMOPP-1) 5,3,4
    3 LPATH = 3
      IF (LPLACE .EQ. ISTEPS) RETURN
    4 LPATH = 2
    5 RETURN
      END
C
      FUNCTION NCHECK(JCH,JCV,JNUM,JCHEKA,JCHEKB)
      COMMON /KSINP/ IDATA(8),IBOARD(8,8,2)
      EQUIVALENCE (IDATA(2),ICOLOR),(IDATA(3),IFROMH),
     1            (IDATA(4),IFROMV),(IDATA(5),ITOH),
     2            (IDATA(6),ITOV)
      DIMENSION ISAVE(6)



C
C FUNCTION NCHECK RETURNS THE NUMBER OF TIMES A KING OF
C COLOR ICOLOR WOULD BE IN CHECK WERE IT ON SQUARE
C (JCH,JCV).  IF JNUM = 1, THE FUNCTION RETURNS AS SOON AS
C THE FIRST CHECK IS FOUND WITH A 1 IN JCHEKA ALSO.  IF
C JNUM = 2, THE SEARCH FOR CHECKS CONTINUES EVEN AFTER THE
C FIRST IS FOUND.  IN THIS CASE, THE TYPES OF THE CHECKS ARE
C RETURNED IN JCHEKA AND JCHEKB, ACCORDING TO THE CODE
C 2 = LONG-DIAGONAL, 3 = SHORT-DIAGONAL, 4 = VERTICAL CHECK,
C 5 = HORIZONTAL CHECK, AND 6 = KNIGHT CHECK.  BOTH VARIABLES
C ARE RETURNED ZEROED IF THERE IS NO CHECK.
C
C THE BASIC METHOD IS TO PLACE A PSEUDA-PIECE ON THE SQUARE
C UNDER TEST, CHANGE THE COLOR OF THE PLAYER TO MOVE, AND
C THEN ATTEMPT TO MOVE EVERY ONE OF THE OPPONENT'S MEN TO
C THE SQUARE IN QUESTION.  SINCE SOME OF THE INPUT TO LDIREC
C IS IN IDATA, SOME PRESENT VALUES ARE SAVED IN ISAVE.  ALSO,
C THE MAN ON (JCH,JCV) IS SAVED.  THE PSEUDO-PIECE HAS VALUE
C SEVEN WITH SIGN ICOLOR.
C
      DO 1 I = 2,6
    1   ISAVE(I) = IDATA(I)
      ISAVE(1) = IBOARD(JCH,JCV,1)
      IBOARD(JCH,JCV,1) = 7*ICOLOR
      ITOH = JCH
      ITOV = JCV
      ICOLOR = -ICOLOR
      NCHECK = 0
C
C IN THE LOOP, OPPONENT'S MEN ARE SEARCHED FOR.  WHEN ONE IS
C FOUND, LDIREC TESTS FOR A LEGAL MOVE.  IF THE MOVE IS
C LEGAL AND JNUM IS 1, A RETURN IS EXECUTED.  IF JNUM IS 2,
C THE TYPE OF CHECK IS CALCULATED.  THE ONLY HARD CASE IS
C THE DISTINCTION BETWEEN LONG AND SHORT DIAGONAL CHECKS.
C THESE ARE HANDLED BY JTEST.  FINALLY, IF THE NUMBER OF
C CHECKS IS TWO, THE LOOP IS LEFT SUMMARILY SINCE A KING MAY
C BE IN CHECK AT MOST TWICE.
      DO 4 IFROMH = 1,8
        DO 4 IFROMV = 1,8
          JMAN = ICOLOR*IBOARD(IFROMH,IFROMV,1)
          IF (JMAN .LE. 0) GOTO 4
          IF (LDIREC(JMAN,JDUMMY) .NE. 1) GOTO 4
          NCHECK = NCHECK + 1
          IF (JNUM .EQ. 2) GOTO 2
          JCHEKA = 1
          GOTO 5
    2     IADELH = IABS(ITOH-IFROMH)
          IADELV = IABS(ITOV-IFROMV)
          IF (IADELH*IADELV .EQ. 2) JCHEK = 6
          IF (IADELV .EQ. 0) JCHEK = 5
          IF (IADELH .EQ. 0) JCHEK = 4
          IF (IADELH .NE. IADELV) GOTO 3
          JCHEK = 3
          JTEST = (9-2*ITOH)*(9-2*ITOV)
          IF (JTEST .GT. 0
     1      .AND. ITOH-IFROMH-ITOV+IFROMV .EQ. 0) JCHEK = 2
          IF (JTEST .LT. 0
     1      .AND. ITOH-IFROMH+ITOV-IFROMV .EQ. 0) JCHEK = 2
    3     IF (NCHECK .EQ. 2) GOTO 7
          JCHEKA = JCHEK
    4     CONTINUE
C
C IF THE LOOP TERMINATES NORMALLY, THERE ARE ZERO OR ONE
C CHECKS, THE BOARD AND IDATA ARE RESET, AND NCHECK EXITS.
C IF THERE WERE TWO CHECKS, STATEMENT 7 SETS JCHEKB AND THEN
C EXITS THE SAME WAY.  IN EITHER CASE, INPUT VALUES OF THE
C BOARD AND IDATA ARE RESTORED BEFORE NCHECK RETURNS.
    5 DO 6 I = 2,6
    6   IDATA(I) = ISAVE(I)
      IBOARD(JCH,JCV,1) = ISAVE(1)
      RETURN
    7 JCHEKB = JCHEK
      GO TO 5
      END
C
      SUBRAUTINE KING(JKH,JKV)
      COMMON /KSINP/ IDATA(8),IBOARD(8,8,2)
      EQUIVALENCE (IDATA(2),ICOLOR)
C
C THE SUBROUTINE KING LOCATES THE KING OF PLAYER ICOLOR AND
C RETURNS THE SQUARE IN (JKH,JKV).  THE KING IS REPPESENTED
C ON THE BOARD WITH THE SAME VALUE AS ICOLOR (I.E., PLUS OR
C MINUS ONE).
C
      DO 1 JKH = 1,8
        DO 1 JKV = 1,8
          IF (IBOARD(JKH,JKV,1) .EQ. ICOLOR) RETURN
    1     CONTINUE
      RETURN
      END
C
      FUNCTION NSIGN(I)
C
C FUNCTION NSIGN RETURNS THE SIGNATURE OF I, WITH SIGNATURE
C OF ZERO EQUAL TO ZERO.
C
      NSIGN = 0
      IF (I) 1,3,2
    1 NSIGN = -1
      RETURN
    2 NSIGN = 1
    3 RETURN
      END
```

Algorithm 90

PERIODIC CUBIC SPLINE INTERPOLATION WITH EQUIDISTANT NODES

W. S. Ford
Department of Electrical Engineering
University of Toronto
Toronto, Ontario
Canada M5S 1A4

Author's note:

A simple algorithm is given for spline interpolation between a number of equi-distant nodes, when periodicity can be assumed; i.e. given co-ordinate values y_i at nodes x_i, $i = 1, 2, \ldots n$, with equal spacing h, a node x_0 is assumed such that $x_i = x_0 + ih$, and $y_0 = y_n$. An interpolating function $S(x) \in C^2[x_0, x_n]$ is then generated, this function being cubic in each sub-interval $[x_{i-1}, x_i]$, with the end constraints $S'(x_0) = S'(x_n)$ and $S''(x_0) = S''(x_n)$.

The approach differs considerably from that in the non-periodic case (Späth, 1969), and the more general periodic cases (Hoskins, King and Andres, 1972; Hoskins and King, 1972), resulting in a significantly simpler algorithm. It also possesses the property that arbitrary accuracy can be achieved, with execution time approximately proportional to the number of correct digits required.

The key equations are derived from the basic theory of cubic splines (Ahlberg, Nilson and Walsh, 1967). The first step involves calculation of the second derivatives or 'moments' of the spline function at the nodes. In the case of equi-distant nodes this involves solution of the following set of equations:

$$\begin{bmatrix} 4 & 1 & 0 & \ldots & 0 & 0 & 1 \\ 1 & 4 & 1 & \ldots & 0 & 0 & 0 \\ 0 & 1 & 4 & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 4 & 1 & 0 \\ 0 & 0 & 0 & \ldots & 1 & 4 & 1 \\ 1 & 0 & 0 & \ldots & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ \vdots \\ \vdots \\ \vdots \\ M_n \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \vdots \\ \vdots \\ \vdots \\ p_n \end{bmatrix}$$

where

$$M_i = (S''(x_i))/6, i = 1, 2, \dots n$$

and

$$p_i = y_{i-1} - 2y_i + y_{i+1}, i = 1, 2, \dots n - 1$$
$$p_0 = p_n = y_{n-1} - 2y_n + y_1 .$$

The inverse of the coefficient matrix is of the form:

$$\begin{bmatrix} \alpha_0 & \alpha_1 & \alpha_2 & \dots & \alpha_3 & \alpha_2 & \alpha_1 \\ \alpha_1 & \alpha_0 & \alpha_1 & \dots & \alpha_4 & \alpha_3 & \alpha_2 \\ \alpha_2 & \alpha_1 & \alpha_0 & \dots & \alpha_5 & \alpha_4 & \alpha_3 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ \alpha_3 & \alpha_4 & \alpha_5 & \dots & \alpha_0 & \alpha_1 & \alpha_2 \\ \alpha_2 & \alpha_3 & \alpha_4 & \dots & \alpha_1 & \alpha_0 & \alpha_1 \\ \alpha_1 & \alpha_2 & \alpha_3 & \dots & \alpha_2 & \alpha_1 & \alpha_0 \end{bmatrix}$$

and as $n \rightarrow \infty$, the values of the α_j approach the limits

$$a_j = (3^{\frac{1}{2}}/2)\,(3^{1/2} - 2)^j, j = 0, 1, 2, \dots$$

The periodic nature of the problem allows it to be expanded to this infinite case with no loss of accuracy, and hence the solution becomes effectively independent of n:

$$M_i = a_0 p_i + a_1(p_{[i-1]} + p_{[i+1]}) + a_2(p_{[i-2]} + p_{[i+2]}) + \dots, \quad i = 1, 2, \dots n$$

where the square brackets [. . .] indicate that the subscripts are reduced modulo n to the range 0, 1, . . . n. The form of the expression for p_i allows this to be expressed:

$$M_i = c_0 y_i + c_1(y_{[i-1]} + y_{[i+1]}) + c_2(y_{[i-2]} + y_{[i+2]}) + \dots, \quad i = 1, 2, \dots n . \quad (1)$$

where

$$c_0 = 2a_1 - 2a_0 = 1 - 3^{1/2}$$
$$c_k = a_{k-1} - 2a_k + a_{k+1} = (2 \times 3^{1/2} - 3)(3^{1/2} - 2)^{k-1}$$
$$k = 1, 2, 3, \dots .$$

The series converges rapidly and arbitrary accuracy can be achieved by using sufficient terms. For greatest accuracy, terms are summed in reverse order. Since successive coefficients alternate in sign and decrease in magnitude by a factor of approximately 13.9 every two terms, a fair estimate of the number of terms required is $2(1 + p - \lfloor p/8 \rfloor)$, where p is the number of decimal digits required to be correct in the computed values of the M_i.

From the function values and second derivatives at two adjacent nodes, the calculation of the interpolating cubic function value at any intervening point is straightforward. For a point x such that $x_{i-1} < x < x_i$, the function can be reduced to the expression:

$$S(x) = S_\delta(\delta) = \delta\{(\delta - 1)(\delta(M_i - M_{i-1}) + M_i + 2M_{i-1}) + (y_i - y_{i-1})\} + y_{i-1}, \; i = 1, 2, \dots n \quad (2)$$

where $\delta = (x - x_{i-1})/h$.

The first derivative of the interpolating function at point x is given by:

$$hS'(x) = S_\delta'(\delta) = (3\delta^2 - 1)(M_i - M_{i-1}) + (6\delta - 3)M_{i-1} + (y_i - y_{i-1}) \quad (3)$$

The algorithm is presented as three procedures. The first uses equation (1) to calculate the node moments for a given n and set of y_i, $i = 1, 2, \dots n$. This procedure could be slightly simplified at the expense of global storage by using a table of precalculated coefficients, rather than calculating them on every call. The other two (function) procedures use equations (2) and (3) respectively to calculate the interpolating function value or its first derivative at any intermediate point.

References

Ahlberg, J. H., Nilson, E. N., and Walsh, J. L. (1967). *The Theory of Splines and Their Applications*, Chapter 2, Academic Press, New York.

Späth, H. (1969). Spline Interpolation of Degree Three (Algorithm 40), *The Computer Journal*, Vol. 12, No. 2, pp. 198-199.

Hoskins, W. D., King, P. R., and Andres, T. H. (1972). Interpolation Using Periodic Splines of Odd Order with Equi-distant Knots (Algorithm 74), *The Computer Journal*, Vol. 15, No. 3, pp. 283-285.

Hoskins, W. D., and King, P. R. (1972). Periodic Cubic Spline Interpolation Using Parametric Splines (Algorithm 73), *The Computer Journal*, Vol. 15, No. 3, pp. 282-283.

```
procedure msolve (y, m, n, p);
  value n, p; integer n, p; array y, m;
comment The co-ordinate values for nodes 1 to n are supplied in
  elements 1 to n of array y[0: n]. This procedure calculates the
  moment values at nodes 0 to n in the corresponding elements of
  array m[0: n]. The number of correct decimal digits required is
  supplied as p;
begin
  real root3, factor, coeff, coeff1;
  integer i, k, kl, kh, kln, khn, nterms;
  y[0] := y[n];
  root3 := sqrt(3);
  nterms := 2 × (1 + p − (p ÷ 8));
  for i := 1 step 1 until n do m[i] := 0;
  coeff1 := 3/(3 + 2 × root3);
  factor := −1/(2 + root3);
  khn := nterms − (nterms ÷ n) × n;
  kln := n − khn;
  for k := nterms − 2 step −1 until 0 do
  begin
    coeff := coeff1 × factor ↑ k;
    khn := if khn = 0 then n − 1 else khn − 1;
    kln := if kln = n then 1 else kln + 1;
    kl := kln; kh := khn;
    for i := n step −1 until 1 do
    begin
      m[i] := m[i] + coeff × (y[kl] + y[kh]);
      kl := if kl = 0 then n − 1 else kl − 1;
      kh := if kh = 0 then n − 1 else kh − 1
    end
  end;
  coeff := −2/(1 + root3);
  for i := 1 step 1 until n do
    m[i] := m[i] + coeff × y[i];
  m[0] := m[n]
end msolve;
real procedure interp (del, i, y, m);
  value del, i; real del; integer i; array y, m;
comment Returns function value at a point x, lying between nodes
  i − 1 and i and defined by del = (x − x[i−1])/h. Procedure msolve
  must previously have been invoked to generate the array m;
interp := del × ((del − 1) × (del × (m[i] − m[i − 1]) + m[i]
  + 2 × m[i − 1]) + (y[i] − y[i − 1])) + y[i − 1];
real procedure slope (del, i, y, m, h);
  value del, i, h; real del, h; integer i; array y, m;
comment As for real procedure interp except returns slope of
  function at point x. Node spacing h must be supplied;
slope := ((3 × del × del − 1) × (m[i] − m[i − 1]) + 3 ×
  (2 × del − 1) × m[i − 1] + (y[i] − y[i − 1]))/h
```

Note on Algorithm 78

COUNTING PREFERENTIAL VOTES (*Vol.* 16, *No.* 3)

Tran van Hoa
Faculty of Economics and Politics
Monash University
Melbourne, Australia

In addition to the points made by I. D. Hill and R. W. M. Wedderburn regarding the function RANF used in the algorithm, and F. Parker's points about misprints in the November 1974 issue (note incidentally that Parker's point 4 is irrelevant, as he has been misled by the printing layout: the '1' of 148H . . . is a continuation character and the Hollerith string has only 48 characters), this algorithm has one very major drawback. It is *wrong*.

There are several mistaken assumptions in its coding.

First, it assumes that all candidates are somebody's first choice. Subroutine EXTR ignores zero elements in the input array, INP. The consequence is that instead of eliminating candidates with no votes before all others, candidates with low non-zero first preference counts are eliminated and their lower preferences may be redistributed to candidates who scored 0 initially. Second, the redistribution of votes is incorrect. In the statements following that labelled 1153 the votes of the candidate being eliminated are transferred to the candidate with lowest preference number on that ballot paper, whether or not the candidate being eliminated is the highest choice remaining on that voting paper. This redistribution must be done only if the candidate being eliminated is the leading preference on that ballot paper and in that case his vote goes to the next preference, *if any*. This means that the whole loop from 5199 to 1150 can be simplified, as one has only to go through the votes once per candidate to transfer the right number.

Third, the redistribution procedure is applied whether or not there are alternative choices on the ballot paper. Subroutine EXTR has to be modified to give JUMP an initial value of 0, which is returned if no element of INP lies in the required range, as is possible when looking at ballot papers after several candidates have been eliminated. There is no compulsion to vote for the required number of candidates or even the right number of vacancies. On exit from EXTR it is then necessary to test the value of JUMP, in case it is 0.

Fourth, as a consequence of the last point, the absolute majority criterion has to be modified on each cycle, to be one more than half the votes still taking part.

It is perhaps worth pointing out that inspection of Tran van Hoa's original data already shows that his subroutine gives false counts. When filling the first vacancy, the votes of candidate A are transferred to candidate F after 6 counts. According to the preference matrix in Table 1 votes 14, 15, and 20 did not vote for candidate F at all and so their voters for other candidates cannot be transferred to him. His final tally should only be 17 instead of 20. Similarly, on the second vacancy counting, when A's votes are transferred to E it is not possible to transfer votes 14 and 20, so that E's maximum vote in this case would be 18. With a different random number generator and the original data I found that candidates were elected in the order F, C, D, A, with final totals of 17, 11, 11 and 20 respectively. This is different from van Hoa's claim of F, E, D, C and also from the manual example of Table 2, which elects in the order A, D, C, B, with 20, 11, 12 and 20 votes.

Perhaps I should add that I found out these things, as well as the points about RANF (especially that it cycled—on unwanted values!) and those noted by Parker and other points on efficiency, the hard way. I tried using the subroutine to count votes in an election, with 11 candidates and 5 vacancies, before their remarks were published. A rewritten version of the subroutine is being submitted separately.

P. A. Samet
Computer Centre
University College London
19 Gordon Street
London WC1H 0AH

Algorithm supplement—Statement of Policy

A contribution to the Supplement may consist of an Algorithm, a Note on a previous algorithm, or an item under the heading of Correspondence.

Algorithms must be submitted in one of the standard programming languages, namely ALGOL 60 (1), ALGOL 68 (2), FORTRAN (3), COBOL (4).

Algorithms in ALGOL 60 or ALGOL 68 must be in form of self-contained procedures. Algorithms in FORTRAN must be self-contained sub-programs. Algorithms in COBOL must be self-contained SUBROUTINEs with parameters or SECTIONs of the PROCEDURE DIVISION and DATA DIVISION. Self-contained means that the algorithm must not use any non-local identifiers other than standard function names, or any COMMON areas. Input/output will normally be through formal parameters, but where standard input/output functions are provided in the language, these are permitted.

The algorithm must be written for publication in the appropriate reference language, and preceded by an appropriate Author's Note. It must be submitted in duplicate and be typewritten double-spaced. Where material is to appear in bold-face it should be underlined in black. Where the appropriate character does not exist on a typewriter, it should be inserted neatly by hand in black and not be replaced by a similar composite character (e.g. $\leqslant$ should not be inserted as $\leq$)

An algorithm must be accompanied by a driver program incorporating it, test data and test results Moreover the tests must be carefully explained on a separate sheet, and all test documentation submitted in duplicate. Test documentation may be in a dialect of the language applicable to a particular computer and have been prepared on the editing equipment of that computer. It may be necessary to ask for paper tape or card source decks in order to check the test documentation independently, but this should not be sent in the first instance.

The Author's Note should include theory of the method, with references, and also explain any tests used to verify the algorithm.

The algorithm which follows the Note should have an opening comment section which briefly defines the parameters used. Comments should be used wherever it is appropriate to clarify the logic. The algorithm must be syntactically correct, produce the results claimed and use computer time and space as efficiently as possible. Avoid constructions whose results may depend on the compiler used (e.g. $y := x + f(x)$ where $f(b)$ is a function which alters the value of b). Cases of failure should be clearly anticipated by the use of appropriate label exits, and commented. Approximate numerical constants must be written as constants and given correct to 15 decimal digits, where appropriate.

Algorithms may be submitted which are translations of published material, but all conditions above must be met. The original Author's permission, where possible, must be obtained in writing and submitted to the Editor, along with the written permission of the copyright holder. Test documentation in such cases must include the original algorithm tests.

Every effort is made to see that published algorithms are completely reliable. In particular all algorithms are submitted to independent referees and extensively checked, so that certifications are not required. However Notes or Correspondence which point out defects in or suggest improvements to previously published algorithms are welcomed. To help in preventing printing mistakes, galley proofs will be sent to authors where possible.

Whilst every effort is made to publish correct algorithms, no liability is assumed by any contributor, the Editor or *The Computer Journal* in connection therewith.

The copyright of all published algorithms remains with *The Computer Journal*. Nevertheless the reproduction of algorithms is explicitly permitted without charge provided that where the algorithm is used in another publication, reference is made to the author and to *The Computer Journal*.

In the event of the formation of a National Algorithm Library, all algorithms which have appeared in the Computer Journal will be made available to this Library.

References

PROGRAMMING LANGUAGE ALGOL, ISO/R/1538.
REPORT ON THE ALGORITHMIC LANGUAGE ALGOL 68. (1969). *Numerische Mathematik*, Vol. 14, pp. 79-218.
PROGRAMMING LANGUAGE FORTRAN, ISO/R/1539.
PROGRAMMING LANGUAGE COBOL, ISO/R/1989.

Documents 1, 3 and 4 above may be obtained from: British Standards Institution, Sales Branch, 101 Pentonville Road, London N.1.

Editor: R. F. Shepherd, Computing Centre, Chelsea College (University of London), Pulton Place, London SW6 5PR.