MATEKBN - Mate with King, Bishop and Knight Practice

Abstract

MATEKBN is a simple BASIC program written in 1980 for the SHARP PC-1211 pocket computer and compatibles to help the user practice the basic checkmate with King, Bishop and Knight vs King within a given number of moves. Two worked examples included.

Keywords: *chess, checkmate, practice, King, Bishop, Knight, KBN vs. K, SHARP PC-1211, compatibles, pocket computer, BASIC*

1. Introduction

MATEKBN is a mid-size *(43 lines, ~1 Kb) BASIC* program that I wrote in 1980 for the *SHARP PC-1211* pocket computer and compatibles (will run *as-is* in the *SHARP PC-1212* and the *TRS-80 PC-1* and with minimal or no changes in other compatible models, see ***Notes 1-2***.)

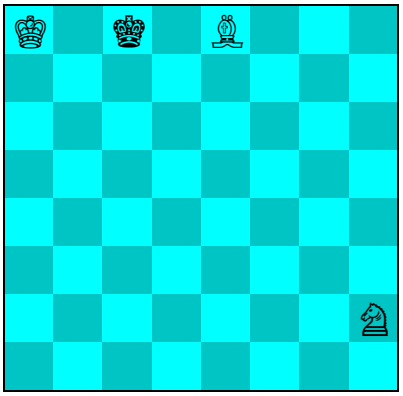
The program is intended to help the user practice in order to achieve the difficult basic checkmate of *King*, *Bishop* and *Knight* (controlled by the user) vs. *King* alone (controlled by the program) within a specified number of moves. The user must try and checkmate before the allotted moves elapse while the program does its best to avoid being checkmated. The possible final outcomes are these:

- the player ***checkmates*** the enemy King within the specified number of moves. It's a ***Win*** for the user.
- the player ***stalemates*** the enemy King within the specified number of moves. The game ends in a ***Draw***.
- the enemy king *captures* the Bishop or the Knight. The game ends in a ***Draw***.
- the specified number of moves *elapses* before any of the above outcomes. The game ends in a ***Draw***.

The program uses a very fast and simple but quite effective positional strategy which will frequently succeed in avoiding being checkmated against beginner and intermediate human players not too experienced with this basic checkmate, and even against old or simple programs not having access to the appropriate endgame tablebase[1].

Except in a negligible number of abnormal initial positions, checkmate can always be achieved in a maximum of 33 moves or less. As per *FIDE* rules, if this mate appears on the board the winning side has a maximum of 50 moves to try and checkmate the lone king, else the game is considered a ***Draw***.

Because of its difficulty and rarity even very strong human players have at times failed to achieve it in serious competition, much to their embarrassment. The program's strategy isn't perfect (only endgame tablebases are guaranteed to play 100% perfect moves for both sides) so once you've trained long enough to be able to beat it in less than 50 moves for most initial positions, try specifying a lower maximum number of moves, say **35** or even **28** for a much harder challenge, little leeway for errors (see ***Examples*** below.) This is one of the longest mates:



*White to play and checkmate in **33***

[1] Of course against an adversary using a KBNK endgame tablebase you can't avoid being checkmated as quickly as possible.

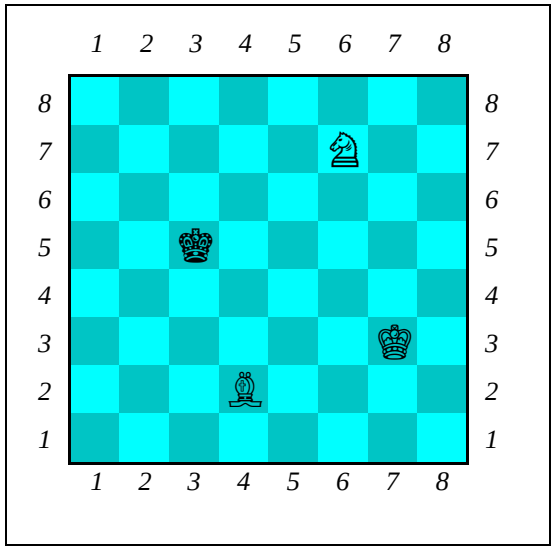
2. Program Listing

```
 1: M=K+11: GOTO 9
 2: M=K+1 : GOTO 9
 3: M=K+10: GOTO 9
 4: M=K-1 : GOTO 9
 5: M=K-10: GOTO 9
 6: M=K-9 : GOTO 9
 7: M=K+9 : GOTO 9
 8: M=K-11
 9: F=1: IF (M<11)+(M>88) RETURN
10: P=M-10*INT(M/10): IF (P=0)+(P=9) RETURN
12: P=ABS(M-R): IF (P=1)+(P=9)+(P=10)+(P=11) RETURN
14: P=ABS(M-C): IF (P=8)+(P=12)+(P=19)+(P=21) RETURN
16: IF M=A LET F=0: RETURN
18: P=M-A: Q=P/11: P=P/9: IF P=INT P LET Q=9*SGN P: GOTO 24
20: IF Q<>INT Q LET F=0: RETURN
22: Q=11*SGN P
24: IF A+Q=M RETURN
26: FOR Z=A+Q TO M-Q STEP Q: IF (Z=C)+(Z=R) LET Z=M: F=0
28: NEXT Z: RETURN
30: "A" INPUT "MY KING IN ";K,"YOUR KING IN ";R,"BISHOP IN ";A,"KNIGHT IN ";C
32: INPUT "MAX. MOVES ? ";N: J=0: USING "###"
34: INPUT "PIECE: K,B,N ? ";P$: IF (P$="K")+(P$="B")+(P$="N")=0 BEEP 2: PAUSE "WRONG PIECE": GOTO 34
36: INPUT "IT MOVES TO ? ";B: GOSUB P$: J=J+1
38: GOSUB (2*(K>50)+(K-10*INT(K/10)>4))*4+56: IF F GOTO 52
40: BEEP 1: K=M: PRINT "#";J; ": I PLAY";K
42: IF M=A BEEP 3: PRINT "BISHOP CAPTURED: DRAW": END
44: IF M=C BEEP 3: PRINT "KNIGHT CAPTURED: DRAW": END
46: IF M=R BEEP 3: PRINT "ILLEGAL KING POSITION": END
48: IF J=N BEEP 3: PRINT N;"-MOVE LIMIT: DRAW": END
50: GOTO 34
52: M=K: GOSUB 9: BEEP 3: IF F PRINT "CHECKMATED IN";J; " MOVES": END
54: PRINT "STALEMATED IN";J;" MOVES": END
56: GOSUB 1: IF F GOSUB 3: IF F GOSUB 2: IF F GOSUB 7: IF F GOSUB 6: IF F GOSUB 5: IF F GOSUB 4: IF F GOSUB 8
58: RETURN
60: GOSUB 7: IF F GOSUB 4: IF F GOSUB 3: IF F GOSUB 8: IF F GOSUB 1: IF F GOSUB 2: IF F GOSUB 5: IF F GOSUB 6
62: RETURN
64: GOSUB 6: IF F GOSUB 2: IF F GOSUB 5: IF F GOSUB 1: IF F GOSUB 8: IF F GOSUB 4: IF F GOSUB 3: IF F GOSUB 7
66: RETURN
68: GOSUB 8: IF F GOSUB 5: IF F GOSUB 4: IF F GOSUB 6: IF F GOSUB 7: IF F GOSUB 3: IF F GOSUB 2: IF F GOSUB 1
70: RETURN
72: "N" C=B: RETURN
74: "B" A=B: RETURN
76: "K" R=B: RETURN
```

***Note**: the boxed line numbers (e.g.: 34:) indicate those lines which are the target of **GOTO** or **GOSUB** branching.*

3. Usage Instructions

The program uses a *column*/*row* convention to specify the coordinates of the square where a piece is currently located or the square where it moves to, where *column* and *row* are single-digit[1] from *1* to *8*. E.g.: in the figure:



- the White King ♔ is located at ***73*** *(col. 7, row 3)*
- the White Bishop ♗ is located at ***42*** *(col. 4, row 2)*
- the White Knight ♘ is located at ***67*** *(col. 6, row 7)*
- the Black King ♚ is located at ***35*** *(col. 3, row 5)*

To run a practice session, in **DEF** Mode proceed as follows:

Step 1: Input the initial position and max. # of moves to mate:

	SHFT **A**	→	*MY KING IN _*
col/row of ♚	**ENTER**	→	*YOUR KING IN _*
col/row of ♔	**ENTER**	→	*BISHOP IN _*
col/row of ♗	**ENTER**	→	*KNIGHT IN _*
col/row of ♘	**ENTER**	→	*MAX. MOVES ? _*
Max. # of moves	**ENTER**	→	*PIECE: K,B,N ? _*

Step 2: Input the piece you're playing and the coordinates *col/row* of the square where you're moving it to:

Very Important: If the initial position is legal, and the user always plays legal moves, the program will *never* play an illegal move either. However, *it doesn't check your moves for legality* so you should make absolutely sure your moves are fully legal before entering them (check for wrong piece or wrong location to move it to) or the current game might get *corrupted* and would have to be amended . See ***Note 3***.

K, B or N[2]	**ENTER**	→	*IT MOVES TO ? _*
col/row	**ENTER**	→	*# 1: I PLAY (col/row) { take note of the program's reply and ... }*
	ENTER	→	*PIECE: K,B,N ? _ { ... press* **ENTER** *to continue }*

Step 3: Repeat *Step 2* above to enter your moves and get the program's replies, until one of these *final* results:

CHECKMATED IN nn MOVES	*You succeeded in giving checkmate within the specified moves. A* ***Win****.*
nn-MOVE LIMIT: DRAW	*You failed in giving checkmate within the specified moves. A* ***Draw****.*
STALEMATED IN nn MOVES	*You stalemated the enemy King. A* ***Draw****.*
BISHOP CAPTURED: DRAW	*The enemy King captured your Bishop. A* ***Draw****.*
KNIGHT CAPTURED: DRAW	*The enemy King captured your Knight. A* ***Draw****.*
ILLEGAL KING POSITION	*You moved your King to an illegal location so it was "captured".* ***Null*** *game.*

Step 4: To practice once more giving checkmate in this same position or a different one, go to *Step 1* above.

---

[1] The standard algebraic notation for chess, where columns are specified by a single *letter* from *a* to *h* instead of a single *digit*, isn't used in this program because the dialect of *BASIC* implemented in the *SHARP PC-1210, PC-1211* and *PC-1212* doesn't include *any* string-handling functions other than assignment, comparison for equality and simple input and output. In particular there's no way to extract characters from a string or convert a character to its ASCII equivalent, so processing inputs such as *e4* or *b7* (let alone *Kb7* or *Bh3*) would be excessively cumbersome and wasteful of memory resources.

[2] If you enter anything other than **K**, **B** or **N** the message *WRONG PIECE* appears and you'll be asked to enter a valid piece.

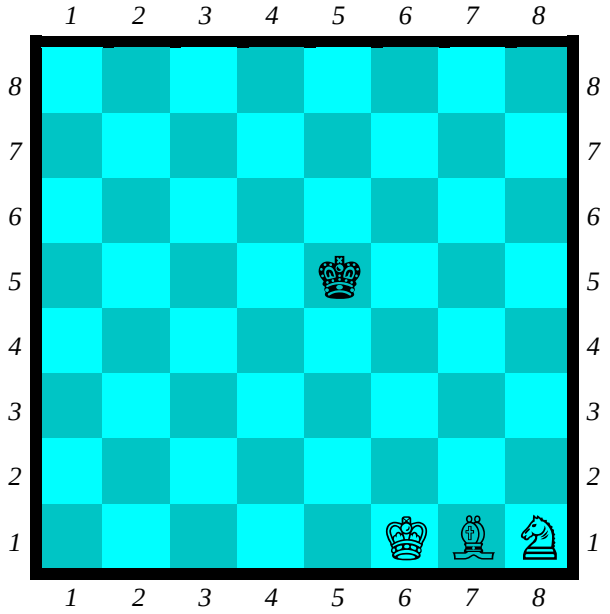
4. Examples

The following examples can be useful to check that the program is correctly entered and to understand its usage.

4.1 Example 1

Playing the White pieces and moving first, try and checkmate the lone Black King starting from this position:

FEN 8/8/8/4k3/8/8/8/5KBN/ w



To conduct the winning forces against our *BASIC* practice program running on a pocket computer released in 1980, the *SHARP PC-1211* (two 4-bit *CPU* @ 256 Khz), we'll use the powerful ***Stockfish 9*** chess engine (> 3,300 ELO, far exceeding Grandmaster level[1]) running at about a million nodes per second on a mid-range *Samsung* tablet, with a hash table of 1 Gbyte (but no endgame tablebases, that would be unfair) and evaluating slightly over 100 million nodes before committing its move.

The given sample position shown left is actually a *Mate in 30* with perfect play by both sides according to the *KBNK Nalimov* tablebase, so let's specify **30** moves as the maximum limit and see how both contenders fare ...

In **DEF** Mode, proceed as follows:

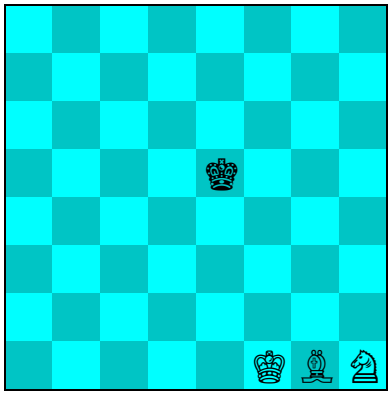
SHFT **A**	→ *MY KING IN _*	*{ remember, positions and moves are entered as column/row }*
55 **ENTER**	→ *YOUR KING IN _*	
61 **ENTER**	→ *BISHOP IN _*	
71 **ENTER**	→ *KNIGHT IN _*	
81 **ENTER**	→ *MAX. MOVES ? _*	
30 **ENTER**	→ *PIECE: K,B,N ? _*	
K **ENTER**	→ *IT MOVES TO ? _*	
52 **ENTER**	→ *# 1: I PLAY 45*	*{ take note of the reply and press* **ENTER** *to continue }*
ENTER	→ *PIECE: K,B,N ? _*	
K **ENTER**	→ *IT MOVES TO ? _*	
43 **ENTER**	→ *# 2: I PLAY 55*	*{* ***Stockfish 9*** *evaluates its move as giving mate in 36 (or less) }*
ENTER	→ *PIECE: K,B,N ? _*	
K **ENTER**	→ *IT MOVES TO ? _*	
34 **ENTER**	→ *# 3: I PLAY 54*	*{* ***Stockfish 9*** *evaluates its move as giving mate in 33 (or less) }*

[1] At the time of writing (2019) all human Grandmasters are rated at less than 3,000 ELO (e.g., Magnus Carlsen: 2,882 ELO)

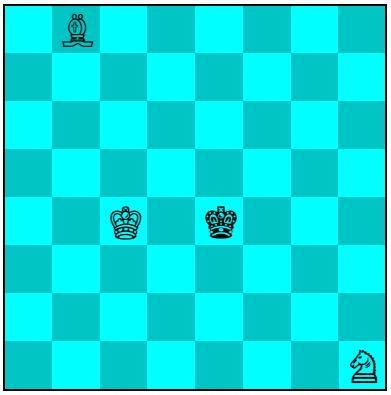
The game goes on with the moves and *SF* evaluations given in the table *(exact Nalimov KBNK eval. in parentheses)*:

#	Stockfish 9	Eval. by SF9	MATEKBN	#	Stockfish 9	Eval. by SF9	MATEKBN
4	*B to 17*	*Mate in 31 (28)*	*I PLAY 55*	*15*	*B to 33*	*Mate in 10*	*I PLAY 31*
5	*B to 28*	*Mate in 29 (27)*	*I PLAY 54*	*16*	*N to 65*	*Mate in 9*	*I PLAY 41*
6	*N to 73*	*Mate in 27 (26)*	*I PLAY 53*	*17*	*N to 53*	*Mate in 8*	*I PLAY 31*
7	*K to 45*	*Mate in 26 (25)*	*I PLAY 43*	*18*	*K to 34*	*Mate in 7*	*I PLAY 21*
8	*B to 17*	*Mate in 17*[1]	*I PLAY 33*	*19*	*K to 23*	*Mate in 6*	*I PLAY 31*
9	*K to 35*	*Mate in 16*	*I PLAY 43*	*20*	*B to 51*	*Mate in 5*	*I PLAY 21*
10	*K to 24*	*Mate in 15*	*I PLAY 42*	*21*	*B to 42*	*Mate in 4*	*I PLAY 11*
11	*K to 34*	*Mate in 14*	*I PLAY 51*	*22*	*N to 32*	*Mate in 3*	*I PLAY 21*
12	*B to 53*	*Mate in 13*	*I PLAY 41*	*23*	*N to 13*	*Mate in 2*	*I PLAY 11*
13	*K to 43*	*Mate in 12*	*I PLAY 51*	*24*	*B to 33*	*Mate in 1*	*CHECKMATED*
14	*B to 44*	*Mate in 11*	*I PLAY 41*				*IN 24 MOVES*

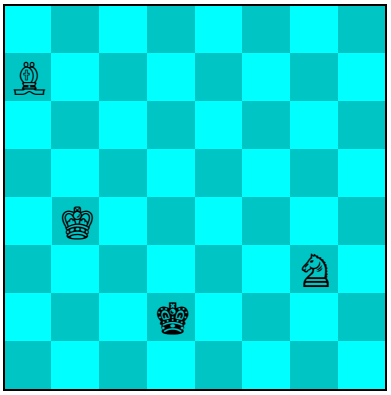
Thus, ***Stockfish 9*** has ***checkmated*** the black King in just **24** moves, less than the specified 30 move limit, but *MATEKBN* resisted bravely and indeed its last 16 moves were *optimal*. Some positions from the game:



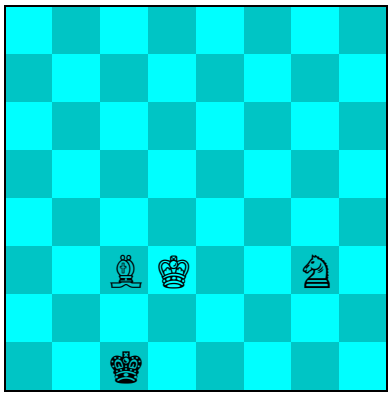
Initial position



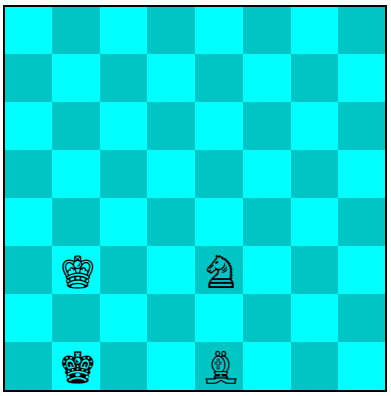
After 5. ... ♔ to 54



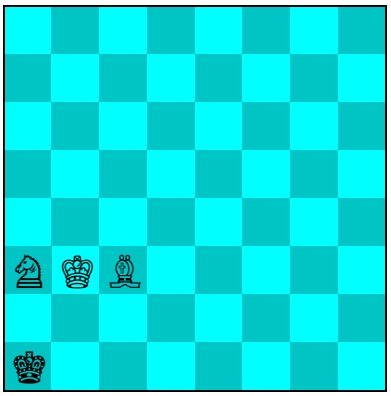
After 10. ... ♔ to 42



After 15. ... ♔ to 31



After 20. ... ♔ to 21



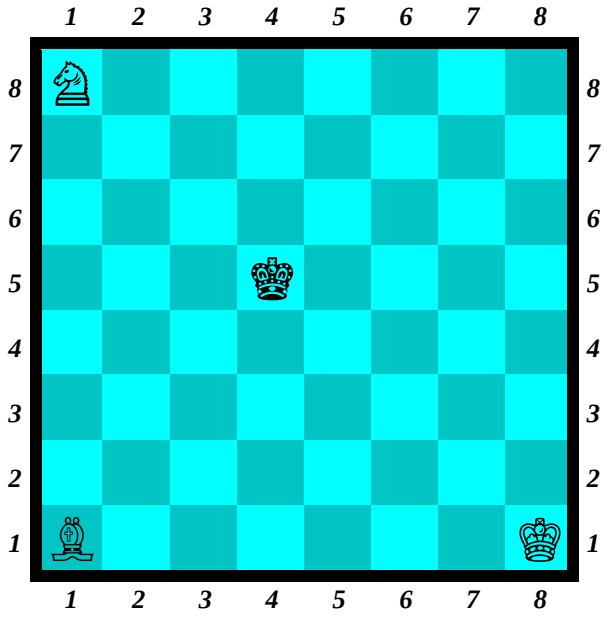
After 24. ♗ to 33++ , checkmate

[1] The previous move #7 by Black was far from optimal, shortening White's future checkmate by no less than 8 moves, but afterwards *all* 16 subsequent Black's moves are 100% perfect and delay the checkmate as much as possible.

4.2 Example 2

Playing the White pieces and moving first, try and checkmate the lone Black King starting from this position:

FEN N7/8/8/3k4/8/8/8/B6K/ w



This time we won't use a powerful program to deliver the mate but we'll let the human user try instead.

The given sample position shown left is also a *Mate in 30* with perfect play by both sides, so we'll specify **35** moves as the maximum limit for the human user to try and checkmate ... or not !

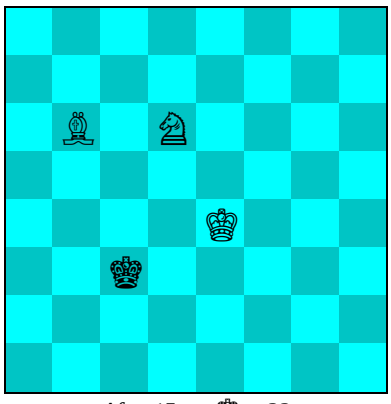
In **DEF** Mode, proceed as follows:

SHFT	**A**	→	*MY KING IN _*
45	**ENTER**	→	*YOUR KING IN _*
81	**ENTER**	→	*BISHOP IN _*
11	**ENTER**	→	*KNIGHT IN _*
18	**ENTER**	→	*MAX. MOVES ? _*
35	**ENTER**	→	*PIECE: K,B,N ? _*

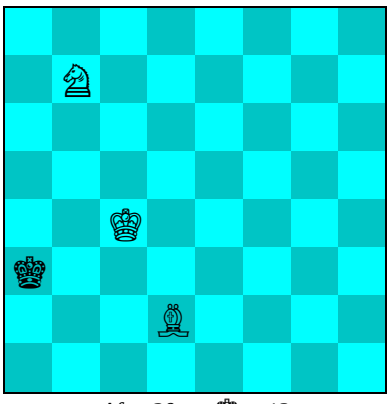
The game goes on with the moves given in the table below:

#	User	MATEKBN	#	User	MATEKBN	#	User	MATEKBN	#	User	MATEKBN
1	*K to 72*	*I PLAY 54*	*8*	*N to 43*	*I PLAY 32*	*15*	*B to 26*	*I PLAY 33*	*22*	*K to 43*	*I PLAY 13*
2	*N to 26*	*I PLAY 43*	*9*	*N to 22*	*I PLAY 23*	*16*	*B to 15*	*I PLAY 23*	*23*	*K to 32*	*I PLAY 12*
3	*K to 63*	*I PLAY 42*	*10*	*K to 43*	*I PLAY 24*	*17*	*K to 43*	*I PLAY 14*	*24*	*B to 24*	*I PLAY 11*
4	*N to 34*	*I PLAY 43*	*11*	*N to 34*	*I PLAY 25*	*18*	*B to 42*	*I PLAY 23*	*25*	*N to 23*	*I PLAY 12*
5	*N to 55*	*I PLAY 42*	*12*	*B to 62*	*I PLAY 24*	*19*	*N to 27*	*I PLAY 14*	*26*	*N to 31*	*I PLAY 11*
6	*B to 44*	*I PLAY 51*	*13*	*N to 46*	*I PLAY 15*	*20*	*K to 34*	*I PLAY 13*	*27*	*B to 13*	*STALEMATED*
7	*K to 53*	*I PLAY 41*	*14*	*K to 54*	*I PLAY 24*	*21*	*N to 35*	*I PLAY 22*			*IN 27 MOVES*

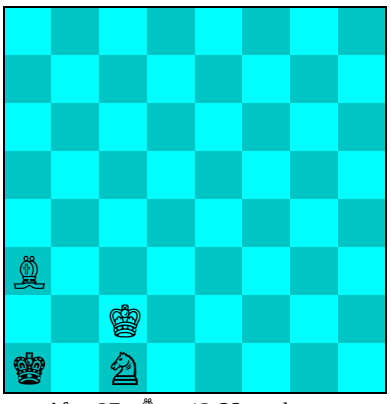
The user's final move was a terrible blunder, ***stalemating*** the Black King, a **Draw**. The correct winning move was ♗ *to 33* to which Black's reply would be *CHECKMATED IN 27 MOVES*. Some positions from the game:



After 15. ... ♔ to 33



After 20. ... ♔ to 13



After 27. ♗ to 13 ??, stalemate

Notes

1. The program can be easily adapted to run under other *BASIC* dialects with minimal or no changes. For example, the only change required for it to run on the *SHARP PC-1350/1360* models is simply to change the usage instructions from "using **DEF** *Mode* and pressing [SHFT] [A] " to "using **RUN** *Mode* and pressing [DEF] [A] ". Also, as these models have a 4-line display instead of the *PC-1211*'s 1-line display, the **PRINT** statement at line *40* can be changed to **PAUSE** and then there would be no need to press [ENTER] to continue.

2. For *BASIC* dialects not supporting *computed* **GOSUB** , the ones at lines *36* and *38* must be substituted by functionally equivalent constructs, like for instance an **IF**..**THEN**..**ELSE** construct for the former and an **ON**..**GOSUB** for the latter (incidentally, none of those constructs are available in *SHARP PC-1211*'s *BASIC*).

3. As stated in the *caveat* at page *3*, if the initial position is legal and the user always plays legal moves, the program will *never* play an illegal move either, but actually *it doesn't check your moves for legality*. If the program happens to play an illegal move, then (program bugs aside for the time being) it's surely the case that either the position is currently illegal, or you played an illegal move, or both. You can check the current locations of all pieces like this: at the > prompt, execute

R [ENTER]	→	shows the current location of the user's *King*	♔
A [ENTER]	→	shows the current location of the user's *Bishop*	♗
C [ENTER]	→	shows the current location of the user's *Knight*	♘
K [ENTER]	→	shows the current location of the program's *King*	♚

If any of these locations are wrong, simply assign the correct locations to the respective variables (say, **R=76** [ENTER]), and resume program execution from line *34* by issuing from the prompt the command [RUN] **34** [ENTER]. This will immediately ask for your move and hopefully the game will then proceed correctly. Else, you'll just have to restart the game.

References

John Nunn (2009)	*Understanding Chess Endgames*
Karsten Müller (2001)	*Fundamental Chess Endings*
Yuri Averbakh (1966)	*Chess Endings: Essential Knowledge*
Yuri Averbach (1979)	*Finales de alfil y de caballo (Spanish)*
Ronald Cohn *et al.* (2012)	*Stockfish*

Copyrights