HP-41C Mate with King, Bishop and Knight Practice

Welcome to a new article, this time starring the ***HP-41C***, the finest combination of functionality, almost infinite expandability and utmost versatility ever seen among RPN calculators, as my *MKBN chess-related* program featured here aptly demonstrates. The *HP42S* has improved hardware specs (speed, *RAM*, display, battery life) and instruction set (complex support, variables, matrices, graphics, menus) but its expandability is ***nil*** (no peripherals but a printer thus no mass storage and no plug-in modules) so it doesn't really compare. The *41C* can be considered the heart of a real system while the *42S* is but a (very) powerful calculator. That's why the *41C* still draws legions of admirers and continues being heavily expanded.

Introduction

MKBN is a mid-size *(285 steps) RPN* program that I wrote in 1980 for the *HP-41C* programmable calculator and compatibles (will run *as-is* in the *HP-41CV* and the *HP-41CX* and with trivial or no changes in other compatible models such as the *HP42S*, see ***Note 1*** and also ***Note 3***.)

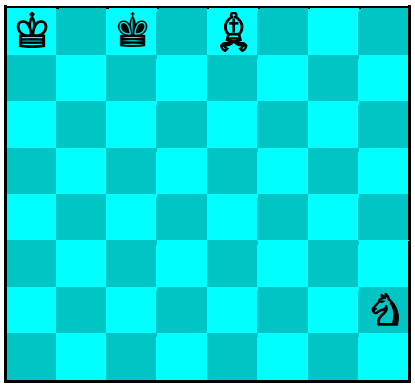
The program is intended to help the user practice in order to achieve the difficult basic checkmate of *King*, *Bishop* and *Knight* (controlled by the user) vs. *King* alone (controlled by the program) within a specified number of moves. The user must try and checkmate before the allotted moves elapse 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* (a few seconds per move) and simple but quite effective positional strategy which will frequently succeed in avoiding being checkmated against human players not too experienced with this basic checkmate, and even against *vintage* programs not having access to the appropriate endgame tablebase[1].

Because of its difficulty and rarity, even very strong human players have at times failed to achieve this checkmate 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.)

Except in a negligible number of abnormal initial positions, checkmate can always be achieved in *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***. 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.

Program Listing for the HP-41C

```
01  LBL "MKBN"
    FIX 0
    CF 29
    CLX
05  "MY KING IN"
    PROMPT
    STO 00
    "YOUR KING IN"
    PROMPT
10  STO 01
    "BISHOP IN"
    PROMPT
    STO 02
    "KNIGHT IN"
15  PROMPT
    STO 03
    "MAX. MOVES ?"
    PROMPT
    STO 04
20  1 E3
    ST/ 04
    ISG 04
    LBL 12
    "YOUR MOVE..."
25  PROMPT
    LBL A
    "KING TO ?"
    PROMPT
    STO 01
30  GTO 00
    LBL B
    "BISHOP TO ?"
    PROMPT
    STO 02
35  GTO 00
    LBL C
    "KNIGHT TO ?"
    PROMPT
    STO 03
40  LBL 00
    RCL 00
    50
    X>Y?
     0
45  25
     /
    14
     +
     5
50  RCL 00
51  10
    MOD
    X>=Y?
    ISG Z
55  LBL 00
    XEQ IND Z
    FS? 00
    GTO 19
    RCL 07
60  STO 00
    ">"
    ARCL 04
    ├": I PLAY "
    ARCL X
65  TONE 9
    PROMPT
    "BISHOP "
    RCL 02
    RCL 07
70  X=Y?
    GTO 09
    "KNIGHT "
    RCL 03
    X=Y?
75  GTO 09
    "ILLEGAL KING"
    ├" POSITION"
    X<>Y
    RCL 01
80  X=Y?
    GTO 11
    CLA
    ARCL 04
    ├" MOVES: DRAW"
85  ISG 04
    GTO 12
    GTO 11
    LBL 09
    ├"CAPTURED: DRAW"
90  GTO 11
    LBL 01
    11
    GTO 00
    LBL 02
95   1
    GTO 00
    LBL 03
    10
    GTO 00
100 LBL 04
101 -1
    GTO 00
    LBL 05
    -10
105 GTO 00
    LBL 06
    -9
    GTO 00
    LBL 07
110  9
    GTO 00
    LBL 08
    -11
    LBL 00
115 RCL 00
     +
    STO 07
    LBL 13
    SF 00
120 11
    RCL 07
    X<Y?
    RTN
    88
125 X<Y?
    RTN
    RCL 07
    10
    MOD
130 X=0?
    RTN
     9
    X=Y?
    RTN
135 RCL 07
    RCL 01
     -
    ABS
     1
140 X=Y?
    RTN
    X<>Y
     9
    X=Y?
145 RTN
    X<>Y
    10
    X=Y?
    RTN
150 X<>Y
151 11
    X=Y?
    RTN
    RCL 07
155 RCL 03
     -
    ABS
     8
    X=Y?
160 RTN
    X<>Y
    12
    X=Y?
    RTN
165 X<>Y
    19
    X=Y?
    RTN
    X<>Y
170 21
    X=Y?
    RTN
    RCL 07
    RCL 02
175 X=Y?
    CF 00
    X=Y?
    RTN
     -
180 STO 09
    11
     /
    STO 10
     9
185 ST/ 09
    RCL 09
    FRC
    X#0?
    GTO 00
190 RCL 09
    X#0?
    SIGN
     9
    GTO 09
195 LBL 00
    RCL 10
    FRC
    X#0?
    CF 00
200 X#0?
201 RTN
    RCL 09
    X#0?
    SIGN
205 11
    LBL 09
     *
    STO 10
    RCL 07
210 RCL 10
    RCL 02
     +
    X=Y?
    RTN
215 STO 08
    RCL 07
    RCL 10
     -
    STO 06
    CF 01
220 RCL 10
    X>0?
    SF 01
    LBL 10
225 RCL 03
    RCL 08
    CF 00
    X=Y?
    RTN
230 RCL 01
    X=Y?
    RTN
    SF 00
    RCL 10
235 ST+ 08
    RCL 06
    RCL 08
    FC? 01
    GTO 00
240 X<=Y?
    GTO 10
    RTN
    LBL 00
    X>=Y?
245 GTO 10
    RTN
    LBL 19
    RCL 00
    STO 07
250 XEQ 13
251 "STALE"
    FS? 00
    "CHECK"
    ├"MATED IN "
255 ARCL 04
    ├" MOVES"
    LBL 11
    AVIEW
    BEEP
260 RTN
    LBL 14
    84567231
    GTO 00
    LBL 15
265 65218347
    GTO 00
    LBL 16
    73481526
    GTO 00
270 LBL 17
    12376458
    LBL 00
    STO 05
    LBL 18
275 RCL 05
    INT
    X=0?
    RTN
    10
280 ST/ 05
    MOD
    XEQ IND X
    FS? 00
    GTO 18
285 END
```

Uses:

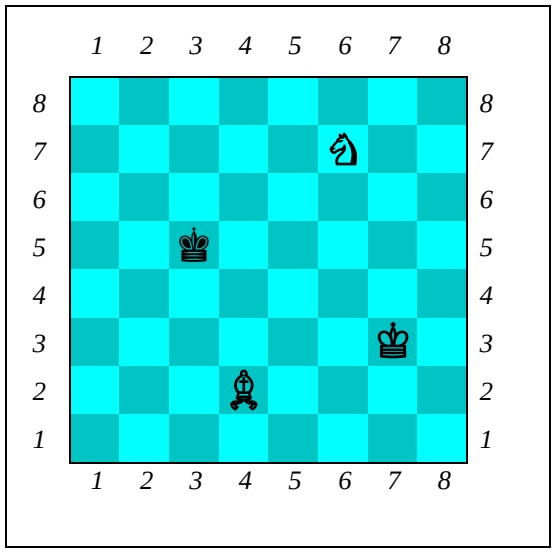
- 285 steps
- sets FIX 0
- flags 0-1, 29
- labels 00-19 and A-C
- registers 00-10

- requires at least 1 Memory Module to run in the original HP-41C.

Note: *to enter text lines use* **ALPHA**, ├ *is the* ***Append*** *character; / is the* ***Divide*** *key and* ******* *is the* ***Multiply*** *key.*

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, proceed as follows:

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

	XEQ	"**MKBN**" →	*MY KING IN*
col/row of ♔	**R/S**	→	*YOUR KING IN*
col/row of ♔	**R/S**	→	*BISHOP IN*
col/row of ♗	**R/S**	→	*KNIGHT IN*
col/row of ♘	**R/S**	→	*MAX. MOVES ?*
Max. # of moves	**R/S**	→	*YOUR MOVE...*

Step 2: In **USER** mode, press A *(King ♔)*, B *(Bishop ♗)* or C *(Knight ♘)* to specify the piece you want to move and the coordinates *col/row* of the square where you're moving it to:

> *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 either your moves or the initial position 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 2***.

A → *KING TO ?* ***or*** B → *BISHOP TO ?* ***or*** C → *KNIGHT TO ?*

col/row **R/S** → *>1: I PLAY (col/row)* *{ take note of the program's reply and press* **R/S** *to continue }*

R/S → *either one of the six possible* ***final results*** *below* ***or*** *YOUR MOVE...*

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 ***MOVES: 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 next to the enemy King so it was "captured". A* ***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 version of *RPN* implemented in the *HP-41C/CV* doesn't include *any* string-handling functions other than storage/recall/append, comparison for equality and simple input/output. In particular there's no simple 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*), while doable, would be cumbersome and wasteful of memory resources.

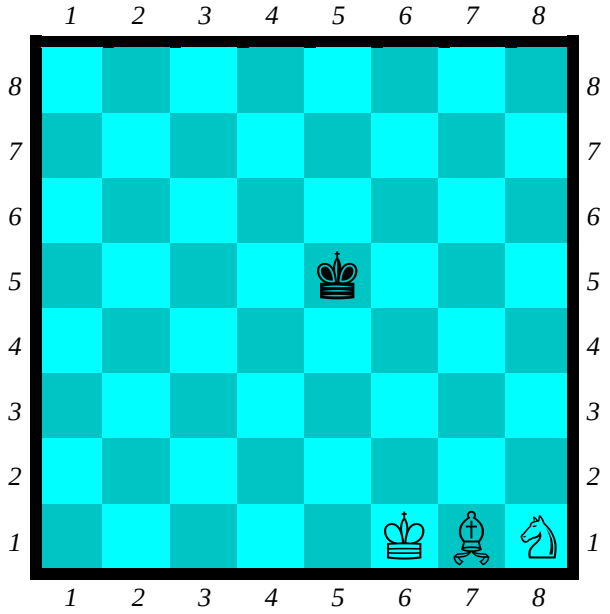
Assorted Examples

Let's see a couple of examples to check that the program is correctly entered and to better understand its usage.

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 *RPN* practice program running on a battery-operated programmable calculator released in 1980, the *HP-41C* (proprietary *CMOS CPU* @ 355 Khz), we'll use the powerful ***Stockfish 9*** *(2018)* 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 **USER** Mode, proceed as follows:

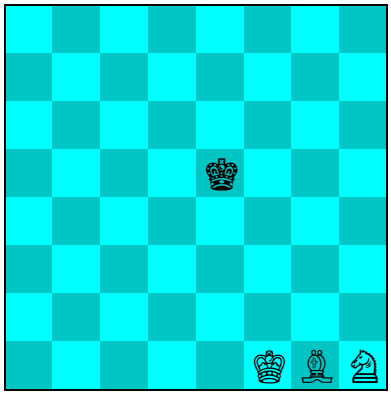
XEQ	"**MKBN**"	→	*MY KING IN*	*{ remember, positions and moves are entered as column/row }*
55	**R/S**	→	*YOUR KING IN*	
61	**R/S**	→	*BISHOP IN*	
71	**R/S**	→	*KNIGHT IN*	
81	**R/S**	→	*MAX. MOVES ?*	
30	**R/S**	→	*YOUR MOVE...*	
	A	→	*KING TO ?*	
52	**R/S**	→	*>1: I PLAY 45*	*{ take note of the reply and **always** press* **R/S** *to continue }*
	R/S	→	*YOUR MOVE...*	
	A	→	*KING TO ?*	
43	**R/S**	→	*>2: I PLAY 55*	*{ **Stockfish 9** evaluates its move as giving mate in 36 (or less) }*
	R/S	→	*YOUR MOVE...*	
	A	→	*KING TO ?*	
34	**R/S**	→	*>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*)

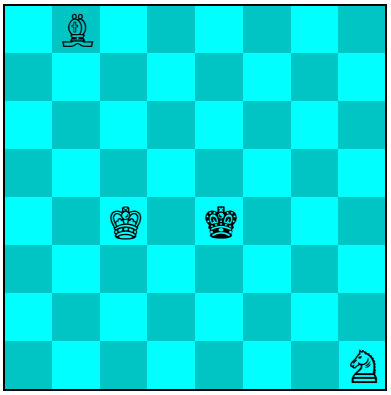
The game goes on with the moves and *SF* evaluations given in the table *(exact Nalimov KBNK eval. in parentheses)*:

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

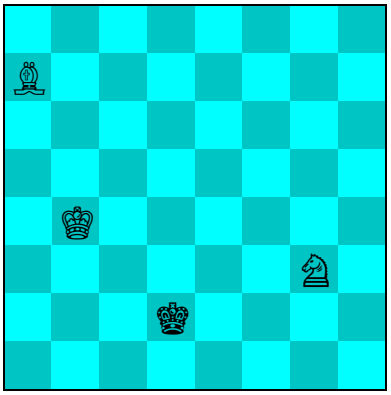
Thus, ***Stockfish 9*** has ***checkmated*** the black King in just **24** moves, less than the specified 30 move limit, but *MKBN* resisted bravely and indeed its last 16 moves were *optimal*. Some positions from the game:



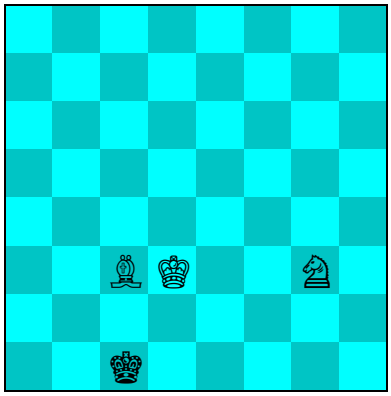
Initial position



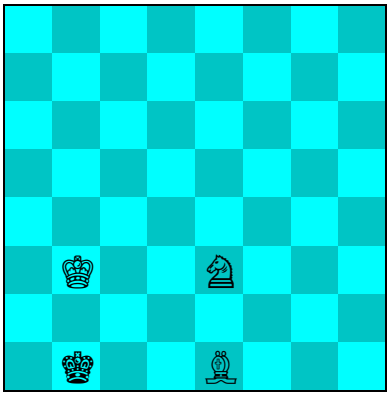
After 5. ... ♕ to 54



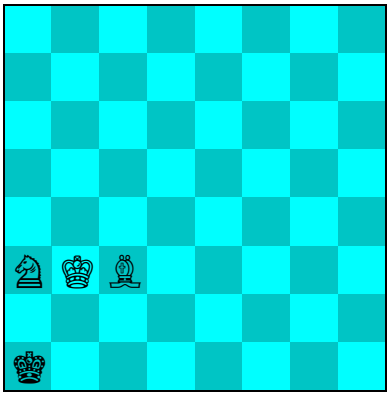
After 10. ... ♕ to 42



After 15. ... ♕ to 31



After 20. ... ♕ to 21



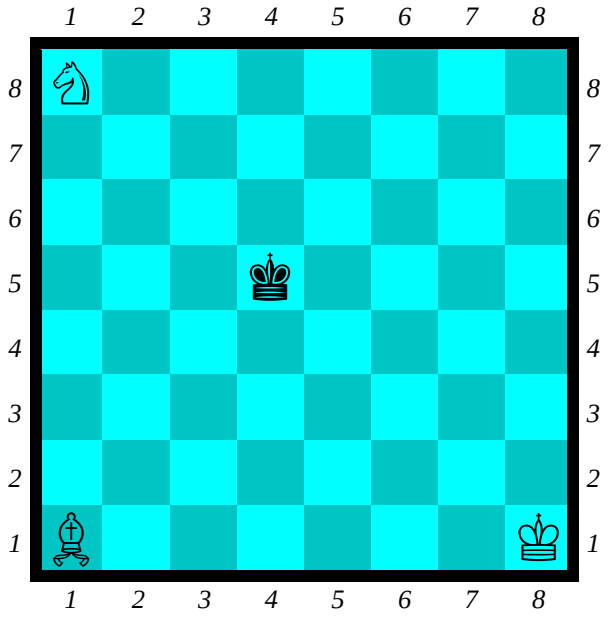
After 24. ♗ to 33++ , checkmate

[1] The previous *move #7* by Black was far from optimal, shortening White's future checkmate by 8 moves, but afterwards *all* 16 subsequent Black's moves are 100% accurate as per *Nalimov KBNK* table and delay the checkmate as much as possible.

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 !

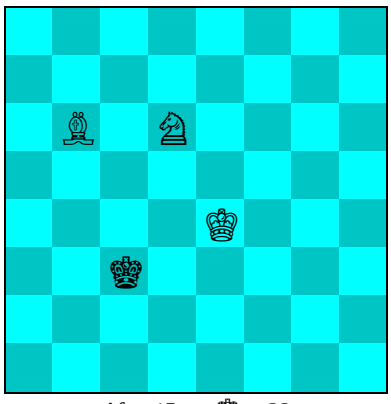
In **USER** Mode, proceed as follows:

XEQ	"**MKBN**"	→	*MY KING IN*
45	**R/S**	→	*YOUR KING IN*
81	**R/S**	→	*BISHOP IN*
11	**R/S**	→	*KNIGHT IN*
18	**R/S**	→	*MAX. MOVES ?*
35	**R/S**	→	*YOUR MOVE...*

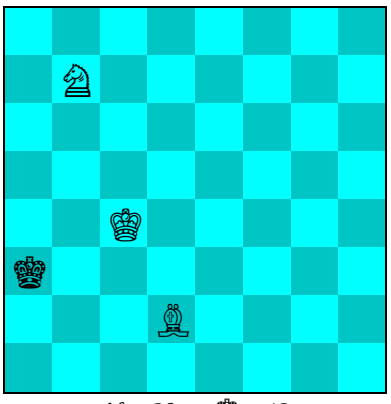
The game goes on with the moves given in the table below:

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

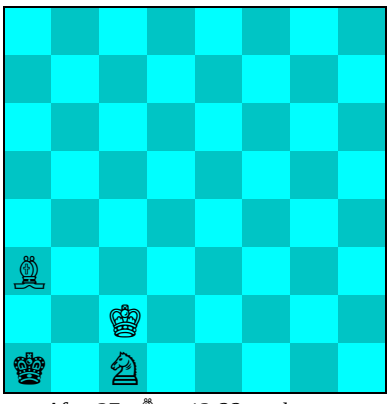
The user's final move was a terrible blunder, ***stalemating*** the Black King, a **Draw**. The correct winning move was ♗ *to 33* , to which Black's reply would be *CHECKMATED IN 27 MOVES*. Some positions from the game:



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 *RPN* versions with minimal or no changes. For example, the only changes required for it to run on the *HP42S* is simply to change the names of some of the intructions such as **FIX 0** to **FIX 00**, ***** to **x**, **/** to **÷**, **ST+/ST-/ST*/ST/** to **STO+/STO-/STOx/STO÷**, respectively, **INT** to **IP**, **FRC** to **FP** , **ISG Z** to **ISG ST Z** and so on. See the ***HP42S Owner's Manual***, in particular the section titled *"Using HP-41C Programs"* for full details. The resulting program size in the *HP42S* should be 643 bytes.

2. 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:

   With the program halted, execute

VIEW 00	→	shows the current location of the program's *King*	♚
VIEW 01	→	shows the current location of the user's *King*	♔
VIEW 02	→	shows the current location of the user's *Bishop*	♗
VIEW 03	→	shows the current location of the user's *Knight*	♘

   If any of these locations happen to be wrong, simply store the correct locations into the respective storage registers (say for instance, if the user's *King* ♔ location is currently wrong and the correct one would be at *column **7*** and *row **6***, then simply execute in **RUN** mode the command: **76** **STO 01**) and once done, resume program execution by issuing the command **XEQ 12** . This will immediately ask for your move and hopefully the game will then proceed correctly. Else, you'll just have to restart the game anew.

Copyrights