UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

School of Computer Science & Applied Mathematics

COMS3005A Assignment
Part 1: State Representation

1 Introduction

In this section, we will develop a way of encoding the *state* of the game. A single state represents the current board position — it should contain all the information necessary for the game (and successor function) to operate correctly.

2 Forsyth–Edwards Notation

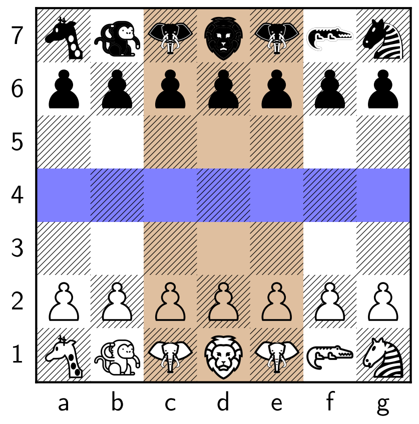
Forsyth–Edwards Notation (FEN) is a standard way of encoding chess positions as a string. For the game Congo, the FEN string will consist of three fields, each separated by a blank space:

```
<position of pieces> <side to move> <move number>
```

2.1 Position of pieces

In our FEN representation, we will specify the placement of each piece on the board. Each piece is encoded as a single character—white pieces uses capital letters, and black pieces use the corresponding lowercase letter. The pieces are: pawn (`P`), giraffe (`G`), monkey (`M`), elephant (`E`), lion (`L`), crocodile (`C`), zebra (`Z`) and superpawn (`S`).

The string describes each rank (row), starting from rank 7 to rank 1. The string for each rank specifices the location of a piece on that rank, and any number of empty squares. Each rank is separated by a `/`. The easiest way to understand this encoding is to look at some examples, as illustrated below.


(a) The starting position with the representation `gmelecz/ppppppp/7/7/7/PPPPPPP/GMELECZ`. This string indicates that rank 7 consists of black pieces only and that they are: giraffe, monkey, elephant, lion, elephant, crocodile, zebra. The next rank is 6, which contains all black pawns. The next rank is 5 which has the representation `7`. This indicates that there are 7 empty squares. The same is true for ranks 4–3. Rank 2 consists of only white pawns, while rank 1 (`GMELECZ`) indicates white giraffe, monkey, elephant, lion, elephant, crocodile, zebra.

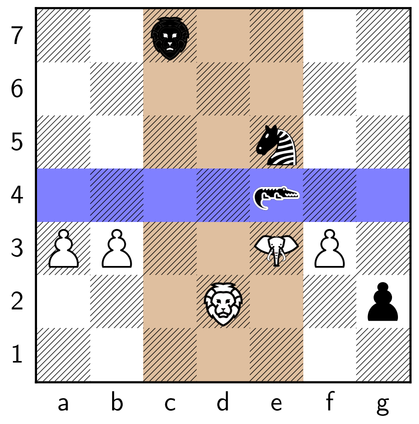

(b) The above board is represented by `2l4/7/4z2/4c2/PP2EP1/3L2p/7`. The 7th rank (`2l4`) has two empty squares, a black lion, and 4 empty squares. The 6th rank has 7 empty squares. The 5th rank (`4z2`) has 4 empty squares, a black zebra and 2 empty squares. The 4th rank (`4c2`) has 4 empty squares, a black crocodile and 2 empty squares. The 3rd rank (`PP2EP1`) has two white pawns, then two empty spaces, a white elephant, a white pawn and an empty space. The 2nd rank (`3L2p`) has 3 empty squares, a white lion, two empty squares and a black pawn. Finally, rank 1 has 7 empty squares.

Figure 1

2.2 Side to move

The side to move is just a single character `W` or `B` that indicates whose turn it is to play

2.3 Move number

The move number records the number of moves played in the game. It is incremented only after black plays a move, and so a value of N indicates that both white and black have made N moves.

1. Position of pieces is a string that specifies the placement of each piece on the board. Each rank is described, starting from rank 7 and ending with rank 1.

Submission: Setting the board

Write a C++ program that accepts a FEN string (as described above), stores the piece location information in appropriate data structures, and then outputs the location of all pieces on the board, as well as the side whose move it is to play.

Input

The first line of input is N, the number of FEN strings that must be read in as input. N lines follow, with each line consisting of a single FEN string.

Hint: a reminder not to be careful when mixing `cin` with `getline`. An example of doing so is below:

```cpp
int N;
cin >> N;
cin.ignore(); //NB!
for (int i = 0; i < N; ++i) {
    string fen;
    getline(cin, fen);
}

```

Output

For each FEN string i, output the location of all pieces on the board, as well as the side to move. Separate each board description with a blank line. Your output should have the following form:

```
white pawn: <square> <square> ...
black pawn: <square> <square> ...
white superpawn: <square> <square> ...
black superpawn: <square> <square> ...
white giraffe: <square> <square> ...
black giraffe: <square> <square> ...
white monkey: <square> <square> ...
black monkey: <square> <square> ...
white elephant: <square> <square> ...
black elephant: <square> <square> ...
white lion: <square> <square> ...
black lion: <square> <square> ...
white crocodile: <square> <square> ...
black crocodile: <square> <square> ...
white zebra: <square> <square> ...
black zebra: <square> <square> ...
side to play: <black|white>
```

where each `<square>` is the file and rank (e.g. `a1`) and squares are separated by blank spaces. If there is more than one square on a line, the squares should be output in alphabetical order.

For example, the string `2l4/7/4z2/4c2/PP2EP1/3L2p/7 b 23` which is illustrated by Figure 1b would produce the output

```
white pawn: a3 b3 f3
black pawn: g2
white superpawn:
black superpawn:
white giraffe:
black giraffe:
white monkey:
black monkey:
white elephant: e3
black elephant:
white lion: d2
black lion: c7
white crocodile:
black crocodile: e4
white zebra:
black zebra: e5
side to play: black
```

Example Input-Output

Sample Input

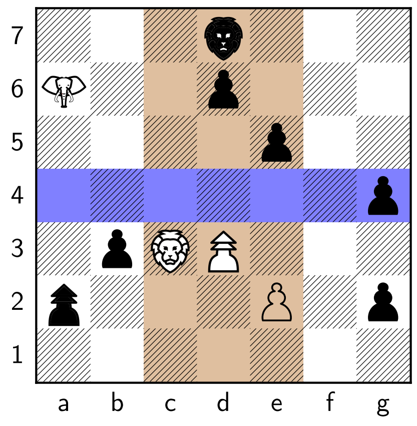
```
2
gmelecz/ppppppp/7/7/7/PPPPPPP/GMELECZ w 0
1m1El2/P1P2P1/1S4C/4S2/1E3S1/1P3c1/2GL3 b 79
```

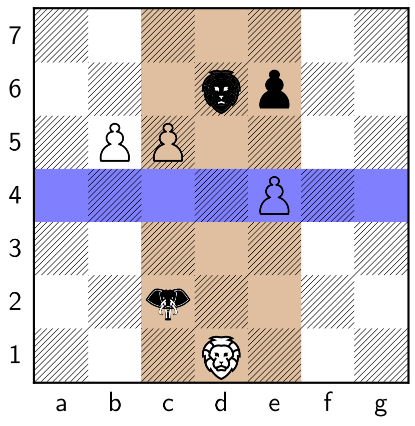
Sample Output

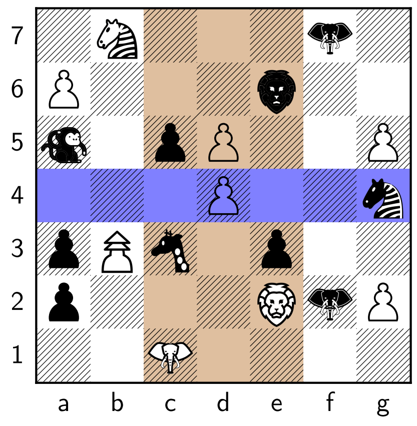
```
white pawn: a2 b2 c2 d2 e2 f2 g2
black pawn: a6 b6 c6 d6 e6 f6 g6
white superpawn:
black superpawn:
white giraffe: a1
black giraffe: a7
white monkey: b1
black monkey: b7
white elephant: c1 e1
black elephant: c7 e7
white lion: d1
black lion: d7
white crocodile: f1
black crocodile: f7
```

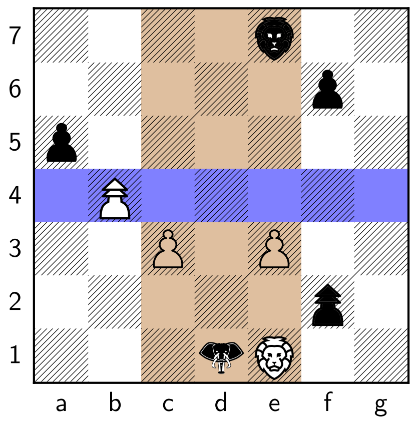
```
white zebra: g1
black zebra: g7
side to play: white

white pawn: a6 b2 c6 f6
black pawn:
white superpawn: b5 e4 f3
black superpawn:
white giraffe: c1
black giraffe:
white monkey:
black monkey: b7
white elephant: b3 d7
black elephant:
white lion: d1
black lion: e7
white crocodile: g5
black crocodile: f2
white zebra:
black zebra:
side to play: black
```

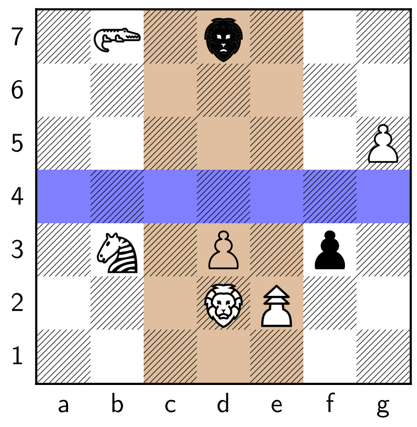
Automatically generated positions

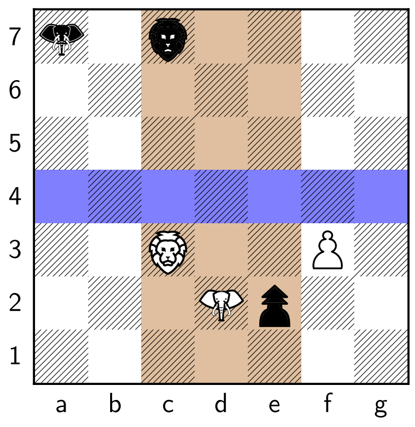

(a) 3l3/E2p3/4p2/6p/1pLS3/s3P1p/7

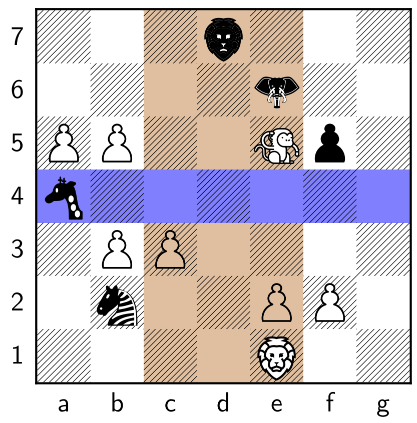

(b) 7/3lp2/1PP4/4P2/7/2e4/3L3

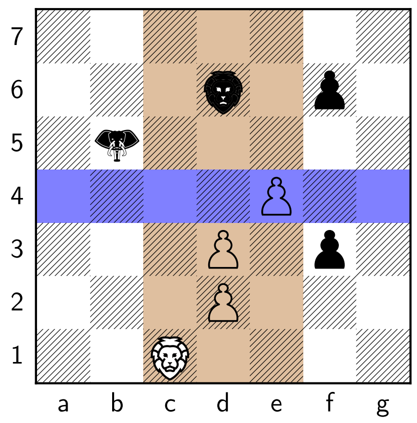

(a) 1Z3e1/P3l2/m1pP2P/3P2z/pSg1p2/p3LeP/2E4

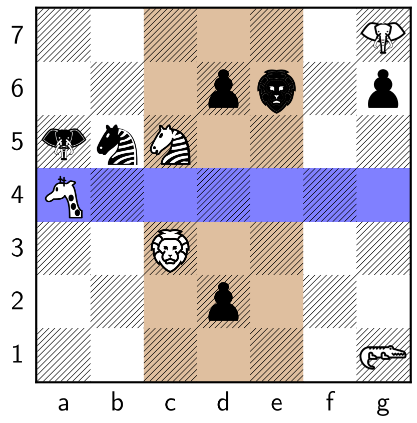

(b) 4l2/5p1/p6/1S5/2P1P2/5s1/3eL2

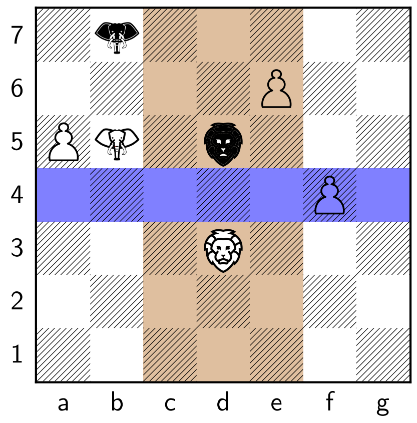

(a) 1C1l3/7/6P/7/1Z1P1p1/3LS2/7

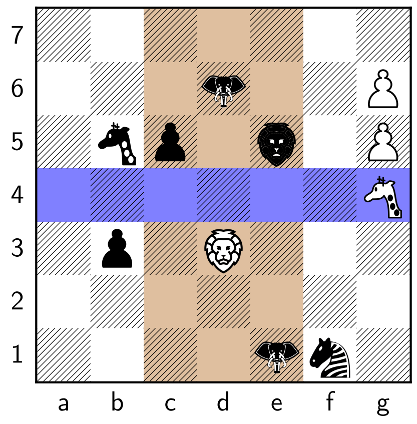

(b) e1l4/7/7/7/2L2P1/3Es2/7

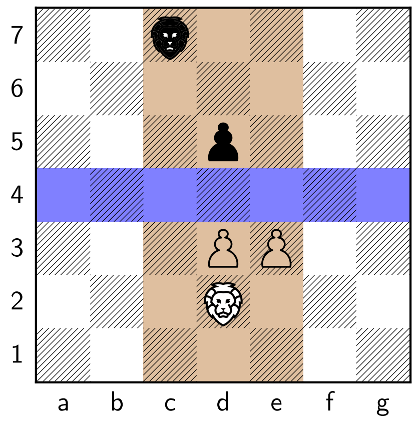

(a) 3l3/4e2/PP2Mp1/g6/1PP4/1z2PP1/4L2

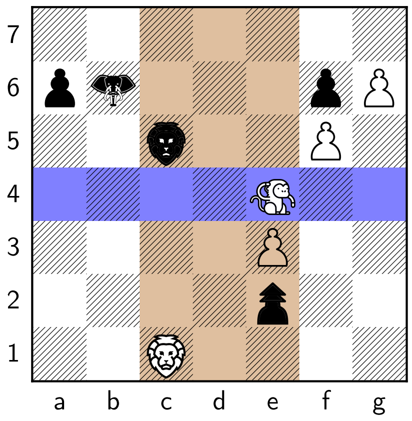

(b) 7/3l1p1/1e5/4P2/3P1p1/3P3/2L4

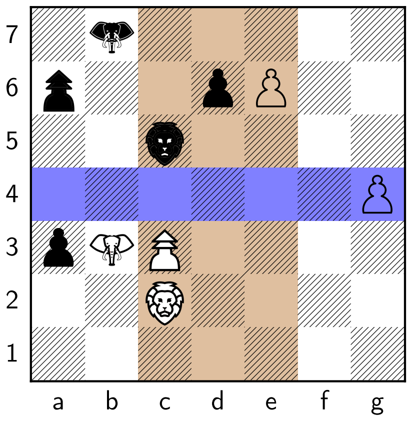

(a) 6E/3pl1p/ezZ4/G6/2L4/3p3/6C

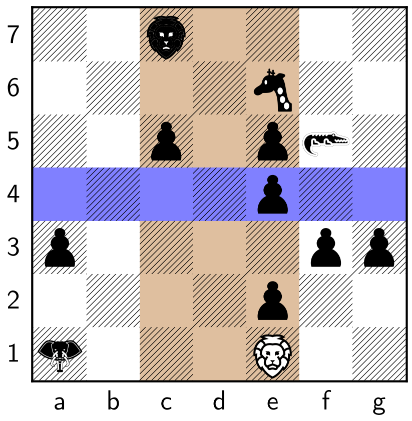

(b) 1e5/4P2/PE1l3/5P1/3L3/7/7

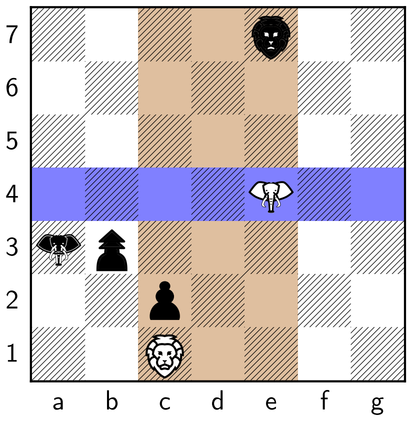

(a) 7/3e2P/1gp1l1P/6G/1p1L3/7/4ez1

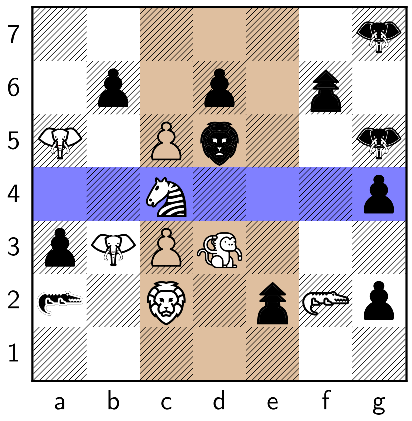

(b) 2l4/7/3p3/7/3PP2/3L3/7

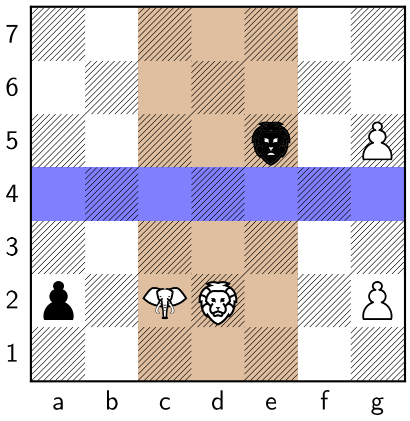

(a) 7/pe3pP/2l2P1/4M2/4P2/4s2/2L4

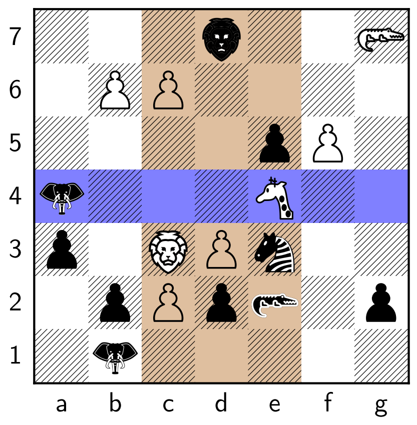

(b) 1e5/s2pP2/2l4/6P/pES4/2L4/7


(a) 2l4/4g2/2p1pc1/4p2/p4pp/4p2/e3L2

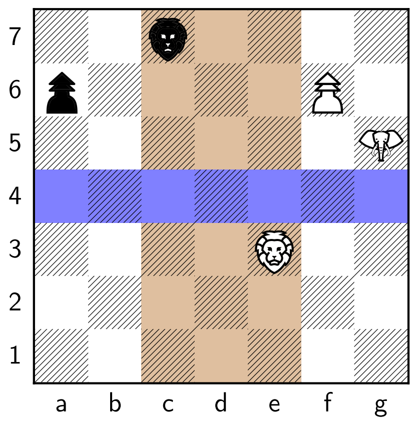

(b) 4l2/7/7/4E2/es5/2p4/2L4


(a) 6e/1p1p1s1/E1Pl2e/2Z3p/pEPM3/c1L1sCp/7

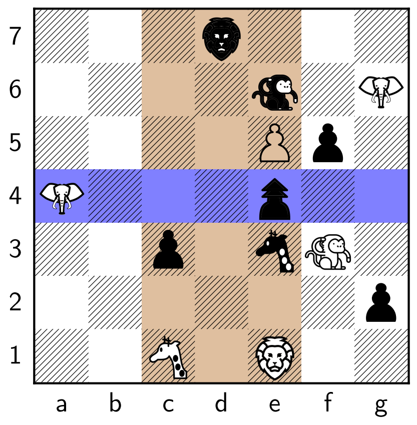

(b) 7/7/4l1P/7/7/p1EL2P/7



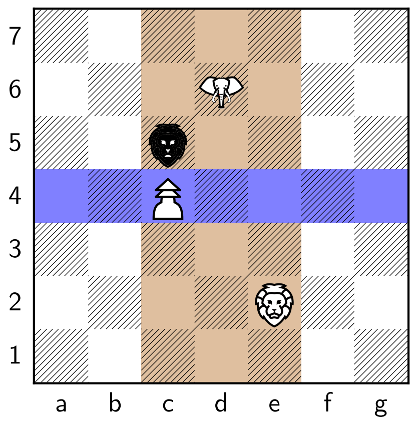
(a) 3l2C/1PP4/4pP1/e3G2/p1LPz2/1pPpc1p/1e5



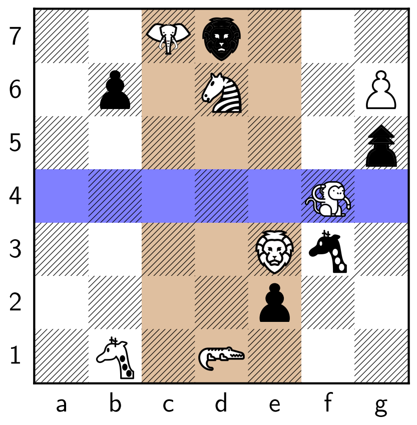
(b) 2l4/s4S1/6E/7/4L2/7/7



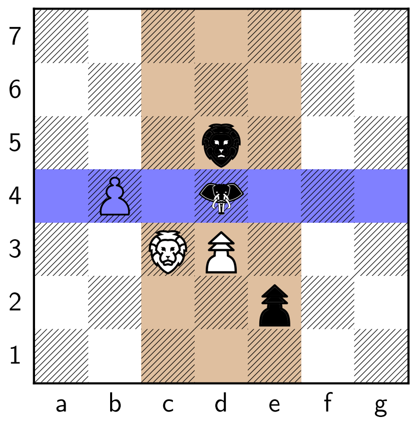
(a) 3l3/4m1E/4Pp1/E3s2/2p1gM1/6p/2G1L2



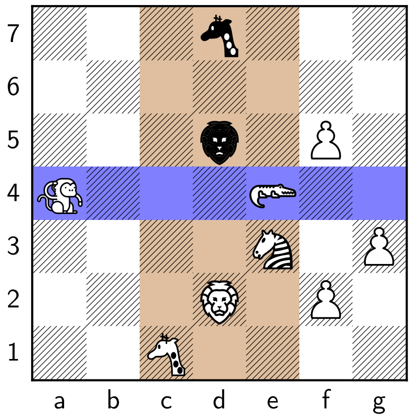
(b) 7/3E3/2l4/2S4/7/4L2/7

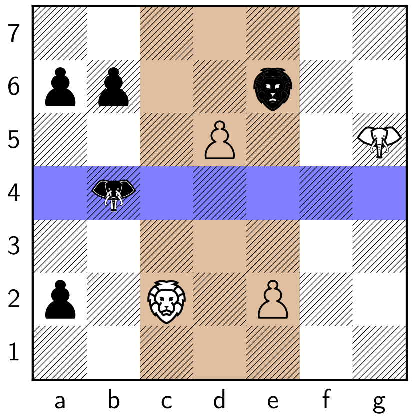


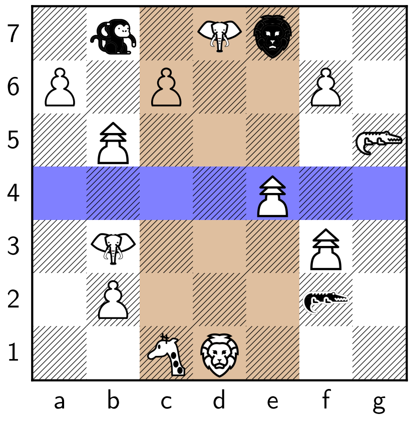
(a) 2El3/1p1Z2P/6s/5M1/4Lg1/4p2/1G1C3

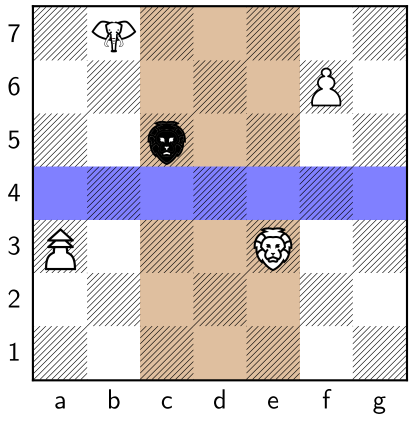


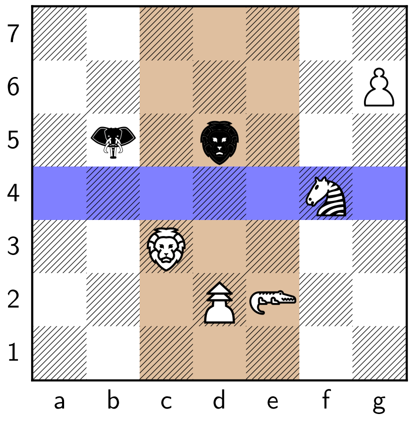
(b) 7/7/3l3/1P1e3/2LS3/4s2/7

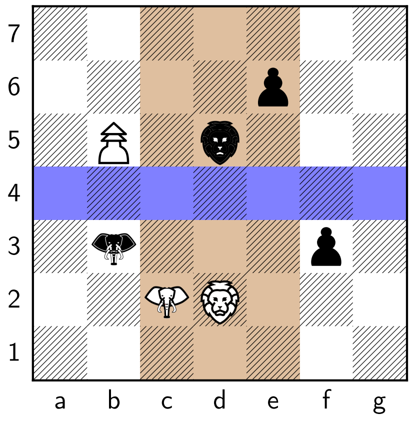

(a) `3g3/7/3l1P1/M3C2/4Z1P/3L1P1/2G4`

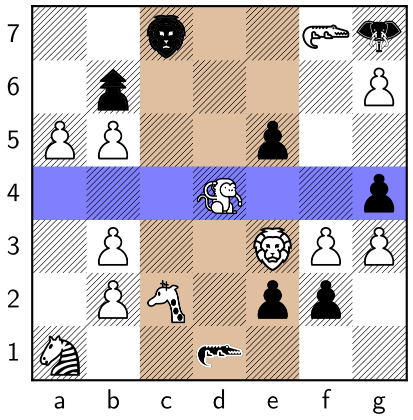

(b) `7/pp2l2/3P2E/1e5/7/p1L1P2/7`


(a) `1m1El2/P1P2P1/1S4C/4S2/1E3S1/1P3c1/2GL3`

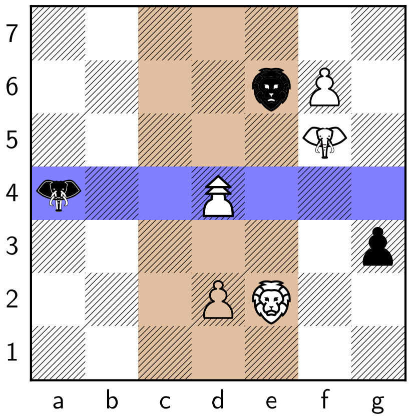

(b) `1E5/5P1/2l4/7/S3L2/7/7`


(a) `7/6P/1e1l3/5Z1/2L4/3SC2/7`

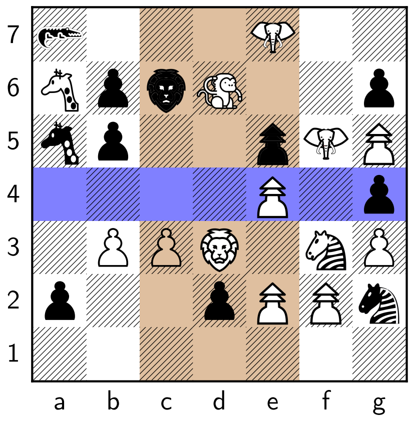

(b) `7/4p2/1S1l3/7/1e3p1/2EL3/7`



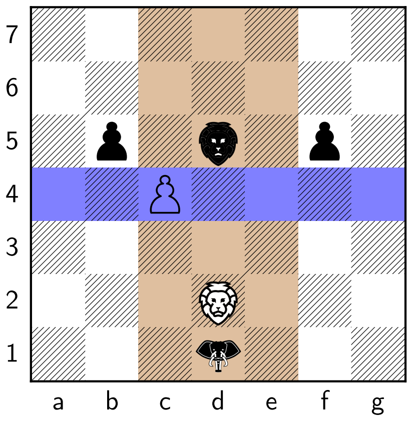
(a) `2l2Ce/1s4P/PP2p2/3M2p/1P2LPP/1PG1pp1/Z2c3`



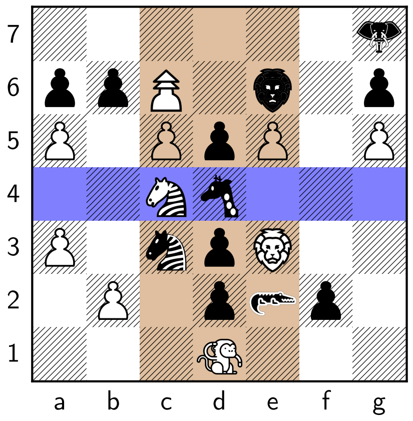
(b) `7/4lP1/5E1/e2S3/6p/3PL2/7`



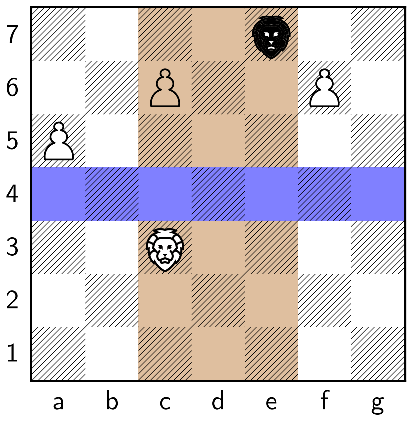
(a) `c3E2/GplM2p/gp2sES/4S1p/1PPL1ZP/p2pSSz/7`



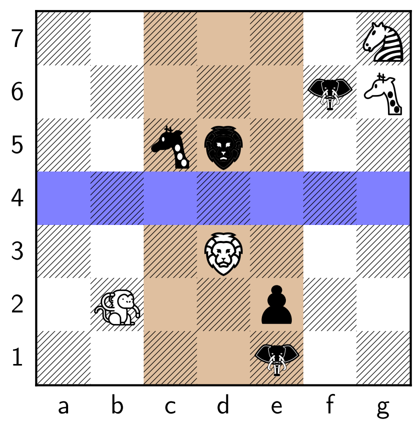
(b) `7/7/1p1l1p1/2P4/7/3L3/3e3`

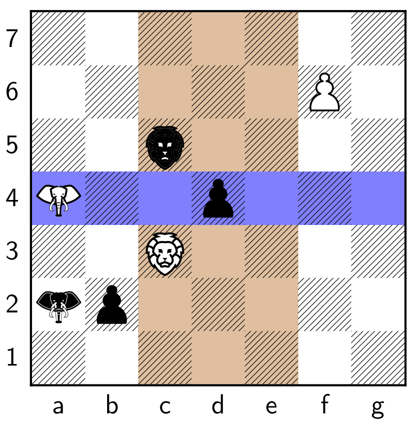


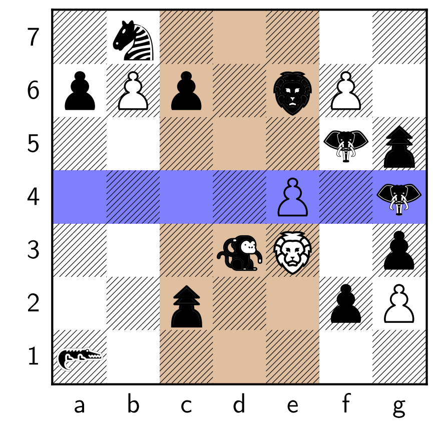
(a) `6e/ppS1l1p/P1PpP1P/2Zg3/P1zpL2/1P1pcp1/3M3`

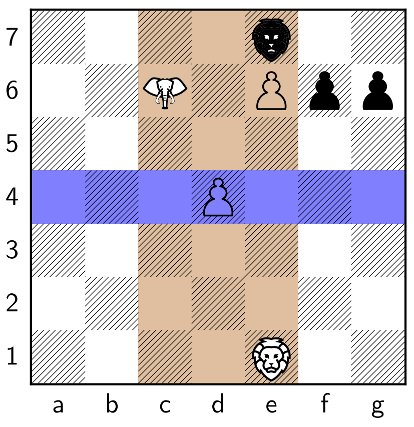


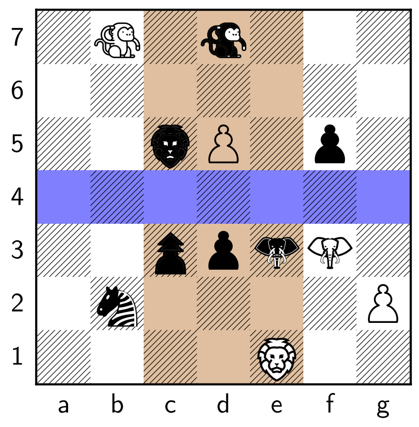
(b) `4l2/2P2P1/P6/7/2L4/7/7`

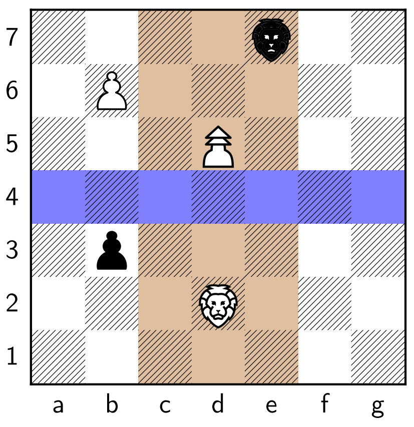

(a) `6Z/5eG/2gl3/7/3L3/1M2p2/4e2`

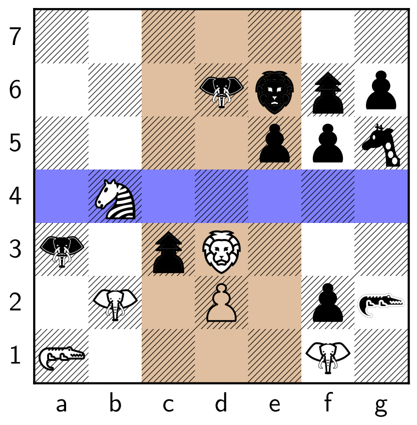

(b) `7/5P1/2l4/E2p3/2L4/ep5/7`

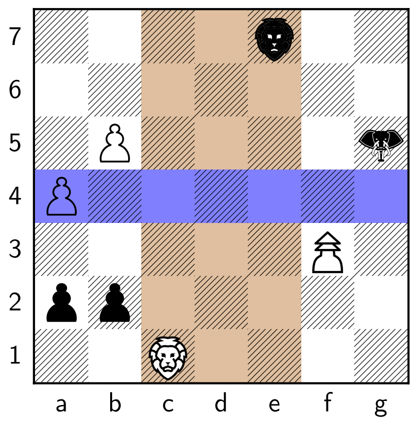

(a) `1z5/pPp1lP1/5es/4P1e/3mL1p/2s2pP/c6`

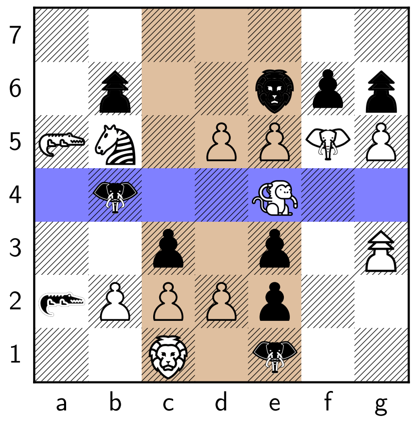

(b) `4l2/2E1Ppp/7/3P3/7/7/4L2`

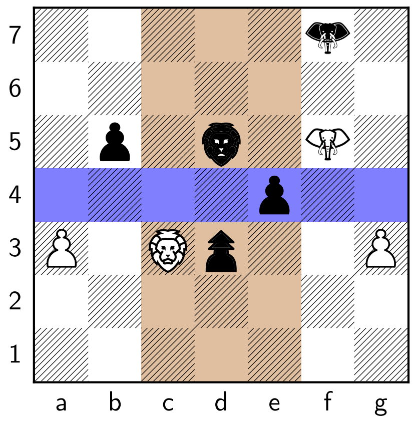

(a) `1M1m3/7/2lP1p1/7/2speE1/1z4P/4L2`

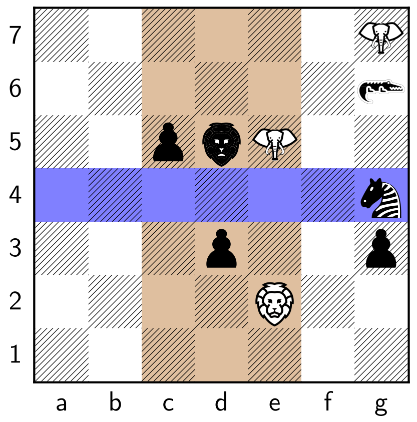

(b) `4l2/1P5/3S3/7/1p5/3L3/7`

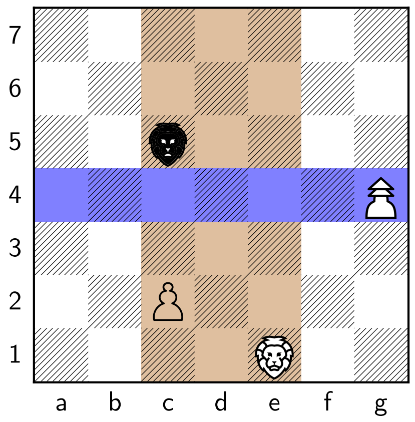

(a) 7/3elsp/4ppg/1Z5/e1sL3/1E1P1pc/C4E1

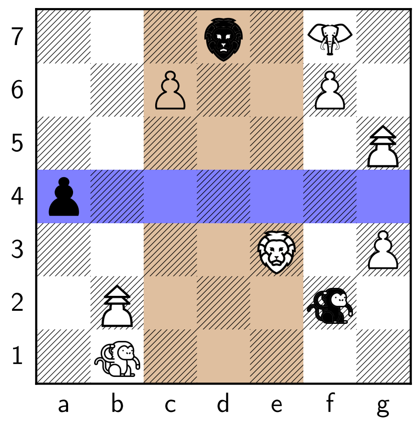

(b) 4l2/7/1P4e/P6/5S1/pp5/2L4


(a) 7/1s2lps/CZ1PPEP/1e2M2/2p1p1S/cPPPp2/2L1e2

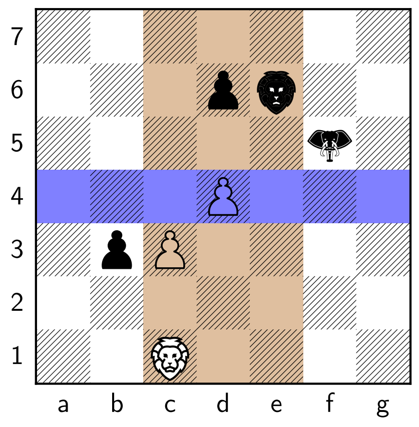

(b) 5e1/7/1p1l1E1/4p2/P1Ls2P/7/7


(a) 6E/6c/2plE2/6z/3p2p/4L2/7

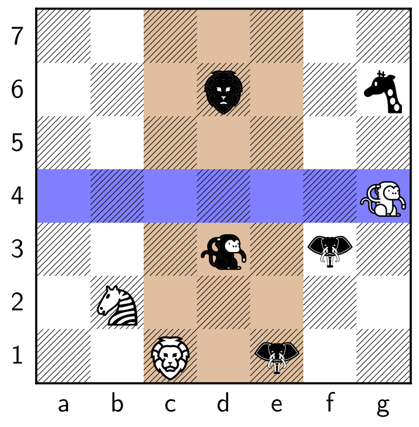

(b) 7/7/2l4/6S/7/2P4/4L2



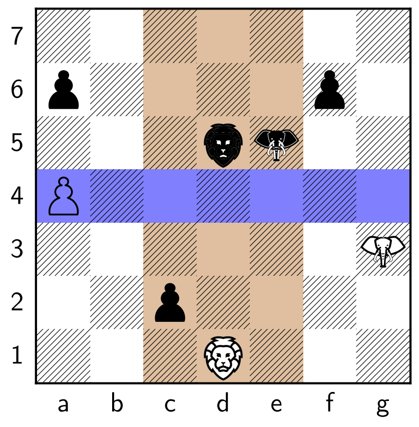
(a) 3l1E1/2P2P1/6S/p6/4L1P/1S3m1/1M5



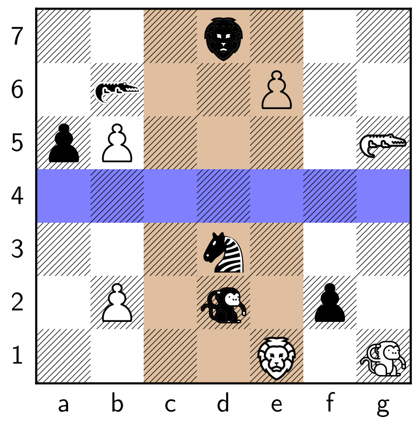
(b) 7/3pl2/5e1/3P3/1pP4/7/2L4



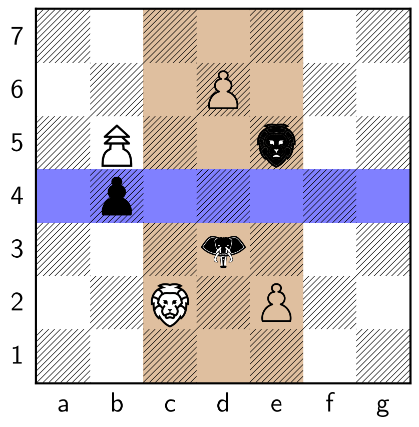
(a) 7/3l2g/7/6M/3m1e1/1Z5/2L1e2



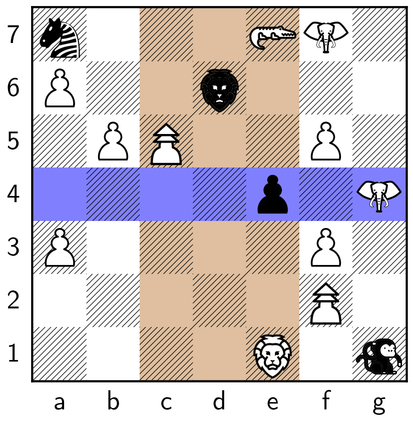
(b) 7/p4p1/3le2/P6/6E/2p4/3L3



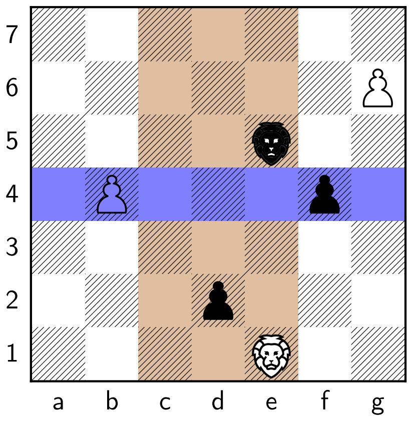
(a) 3l3/1c2P2/pP4C/7/3z3/1P1m1p1/4L1M



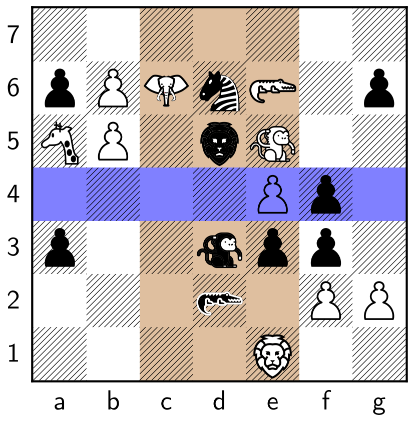
(b) 7/3P3/1S2l2/1p5/3e3/2L1P2/7



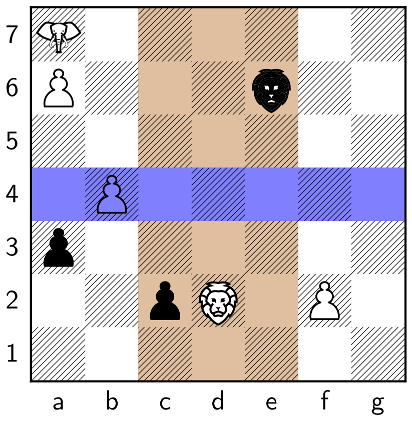
(a) `z3CE1/P2l3/1PS2P1/4p1E/P4P1/5S1/4L1m`



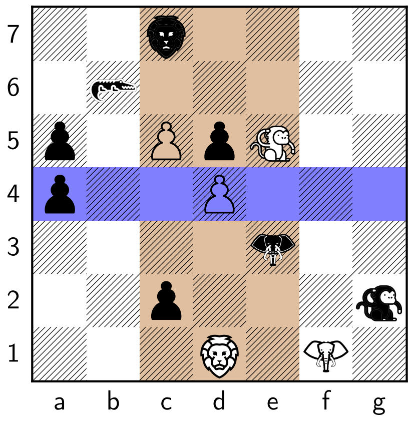
(b) `7/6P/4l2/1P3p1/7/3p3/4L2`



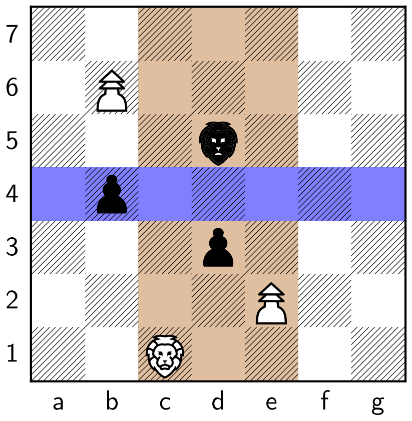
(a) `7/pPEzC1p/GP1lM2/4Pp1/p2mpp1/3c1PP/4L2`



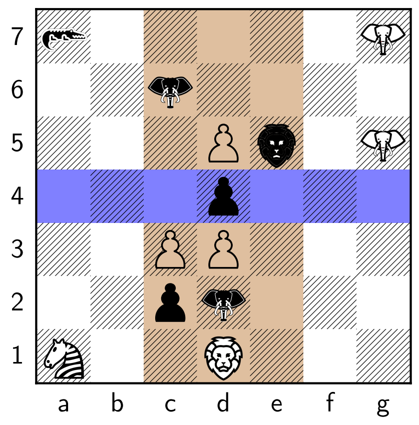
(b) `E6/P3l2/7/1P5/p6/2pL1P1/7`

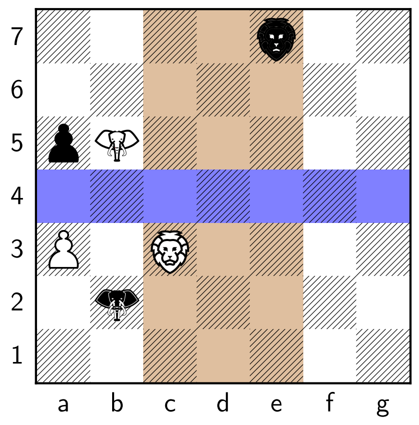


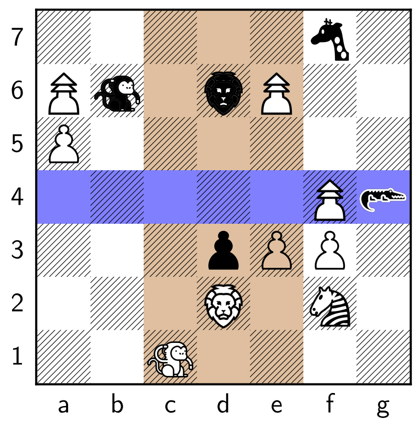
(a) `2l4/1c5/p1PpM2/p2P3/4e2/2p3m/3L1E1`

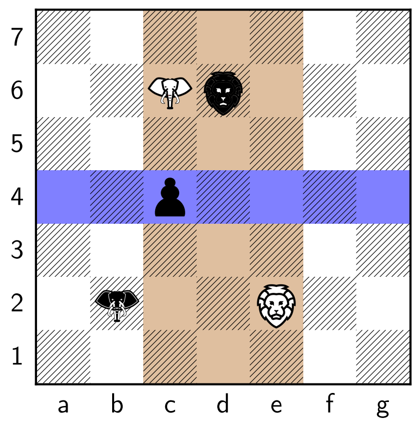


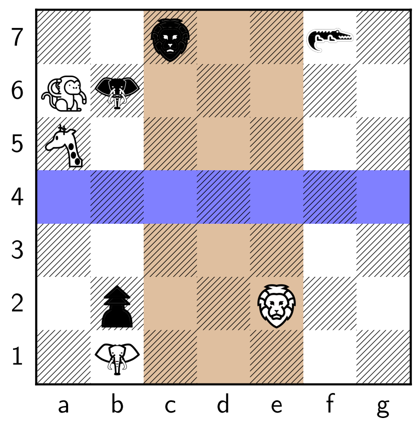
(b) `7/1S5/3l3/1p5/3p3/4S2/2L4`

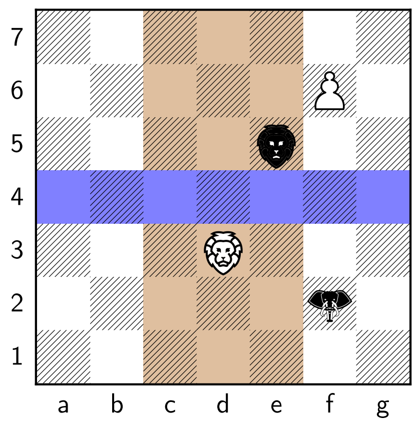

(a) `c5E/2e4/3Pl1E/3p3/2PP3/2pe3/Z2L3`

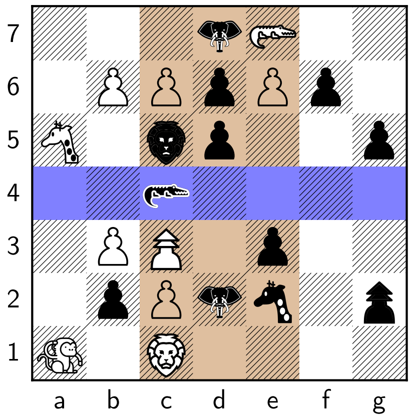

(b) `4l2/7/pE5/7/P1L4/1e5/7`

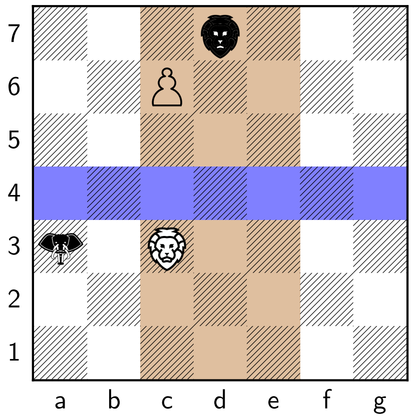

(a) `5g1/Sm1lS2/P6/5Sc/3pPP1/3L1Z1/2M4`

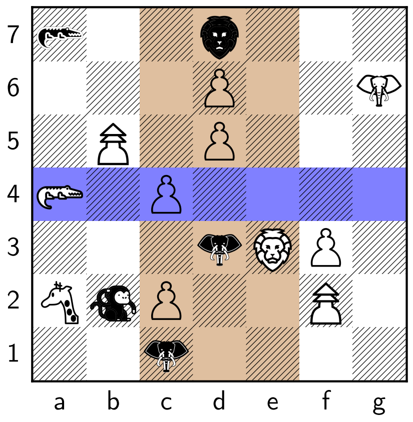

(b) `7/2El3/7/2p4/7/1e2L2/7`

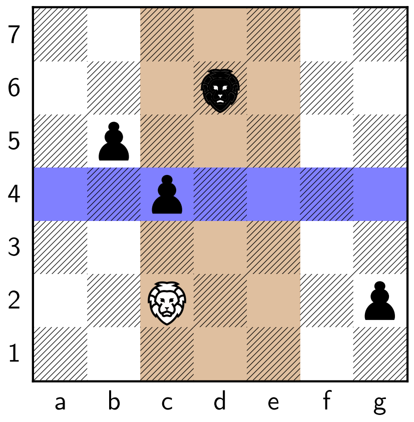

(a) `2l2c1/Me5/G6/7/7/1s2L2/1E5`

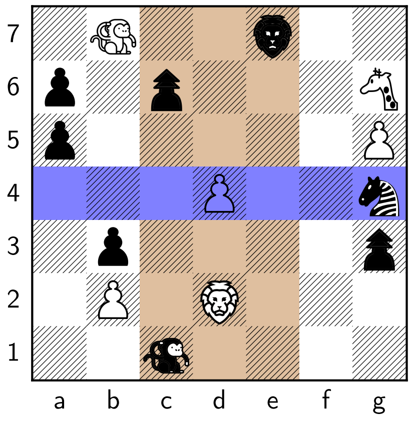

(b) `7/5P1/4l2/7/3L3/5e1/7`

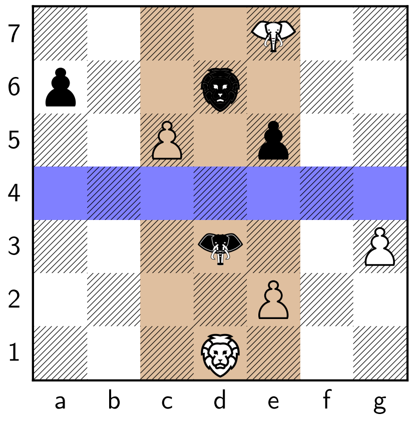

(a) 3eC2/1PPpPp1/G1lp2p/2c4/1PS1p2/1pPeg1s/M1L4

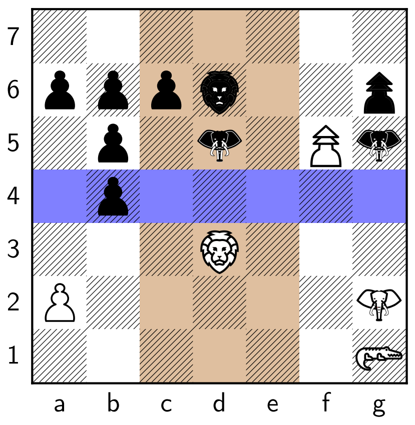

(b) 3l3/2P4/7/7/e1L4/7/7


(a) c2l3/3P2E/1S1P3/C1P4/3eLP1/GmP2S1/2e4

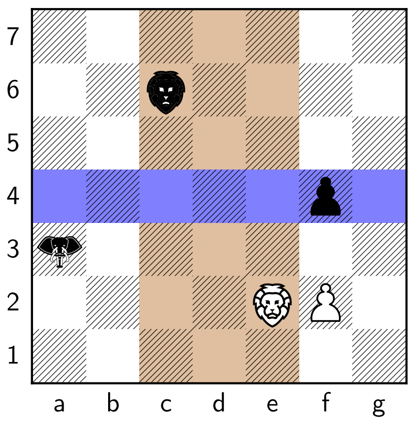

(b) 7/3l3/1p5/2p4/7/2L3p/7


(a) 1M2l2/p1s3G/p5P/3P2z/1p4s/1P1L3/2m4

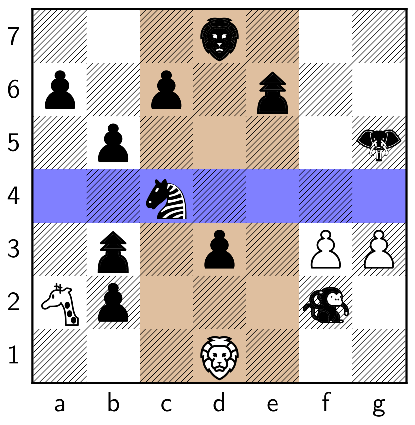

(b) 4E2/p2l3/2P1p2/7/3e2P/4P2/3L3



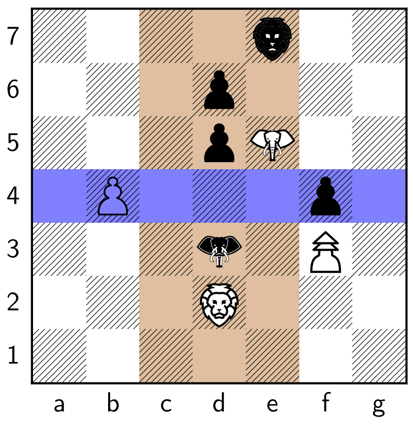
(a) `7/pppl2s/1p1e1Se/1p5/3L3/P5E/6C`



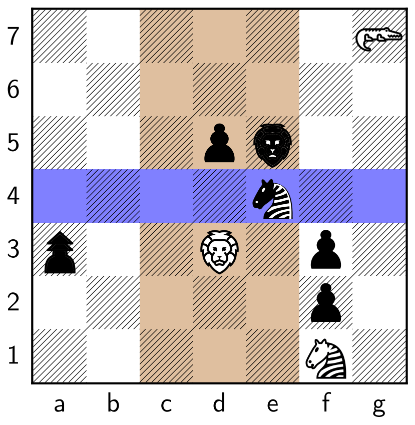
(b) `7/2l4/7/5p1/e6/4LP1/7`



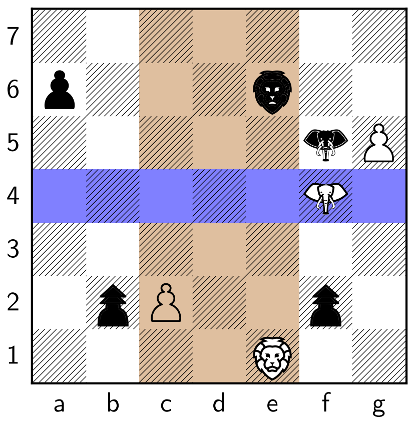
(a) `3l3/p1p1s2/1p4e/2z4/1s1p1PP/Gp3m1/3L3`



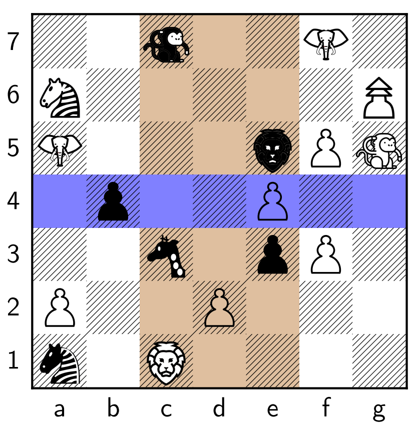
(b) `4l2/3p3/3pE2/1P3p1/3e1S1/3L3/7`

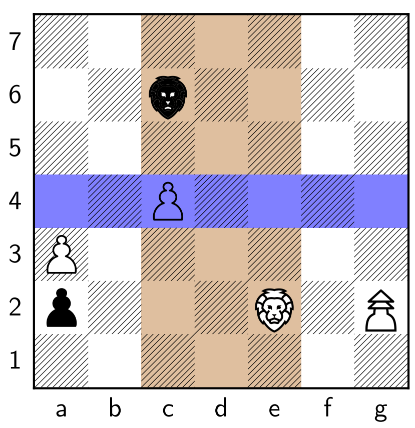


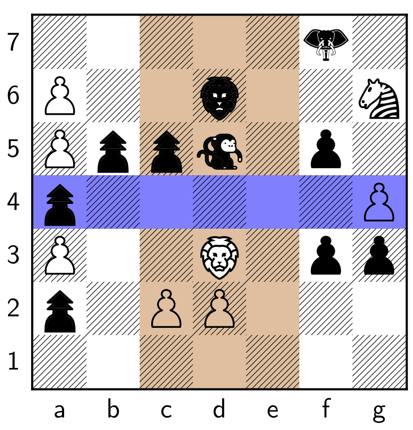
(a) `6C/7/3pl2/4z2/s2L1p1/5p1/5Z1`

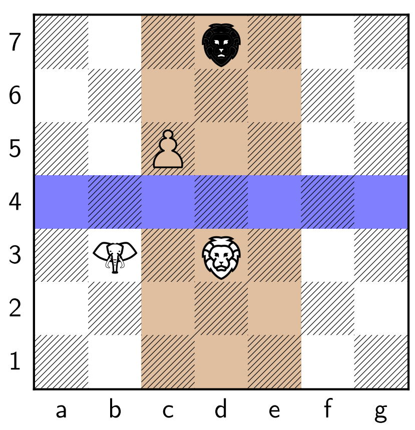


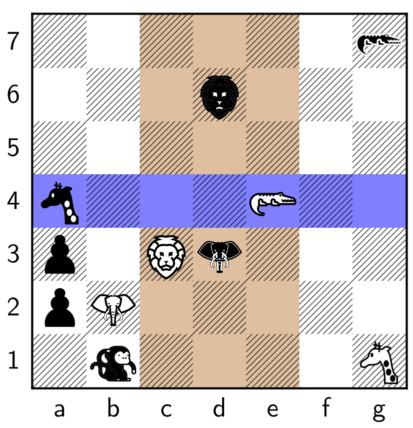
(b) `7/p3l2/5eP/5E1/7/1sP2s1/4L2`

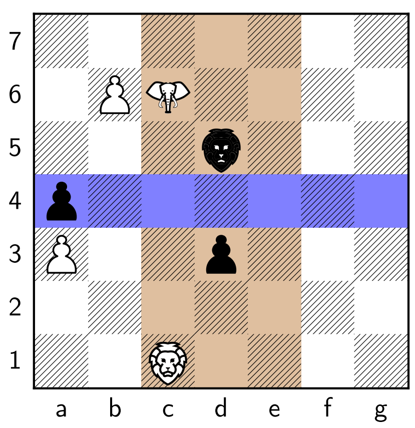

(a) `2m2E1/Z5S/E3lPM/1p2P2/2g1pP1/P2P3/z1L4`

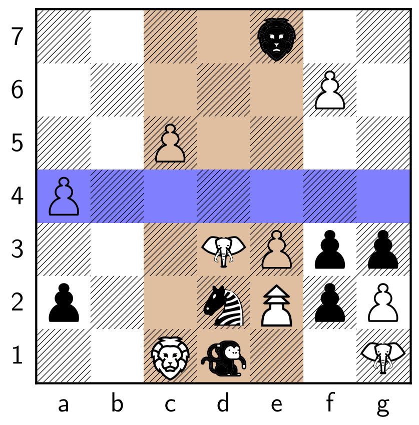

(b) `7/2l4/7/2P4/P6/p3L1S/7`


(a) `5e1/P2l2Z/Pssm1p1/s5P/P2L1pp/s1PP3/7`

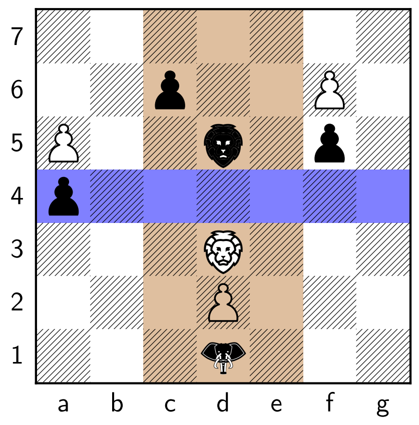

(b) `3l3/7/2P4/7/1E1L3/7/7`


(a) `6c/3l3/7/g3C2/p1Le3/pE5/1m4G`

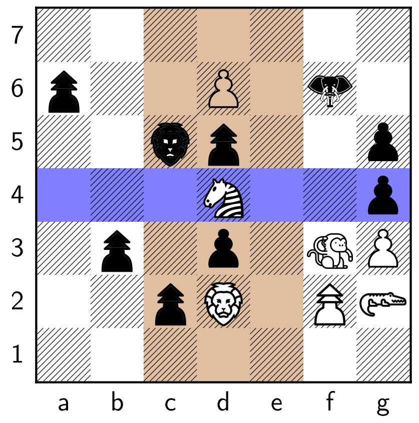

(b) `7/1PE4/3l3/p6/P2p3/7/2L4`



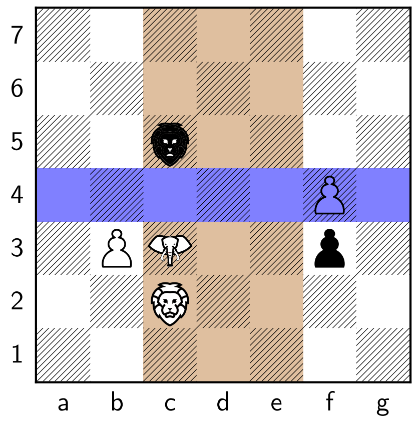
(a) `4l2/5P1/2P4/P6/3EPpp/p2zSpP/2Lm2E`



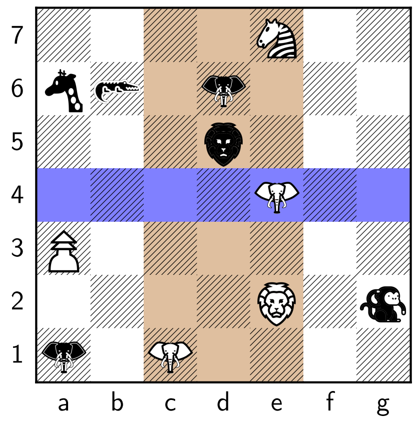
(b) `7/2p2P1/P2l1p1/p6/3L3/3P3/3e3`



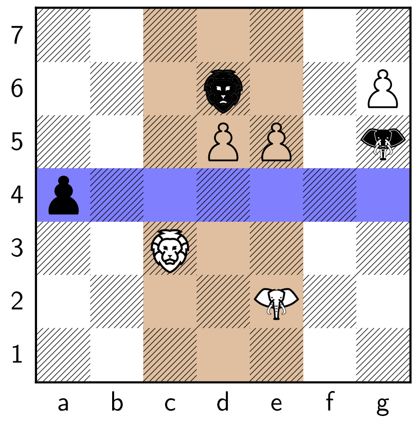
(a) `7/s2P1e1/2ls2p/3Z2p/1s1p1MP/2sL1SC/7`



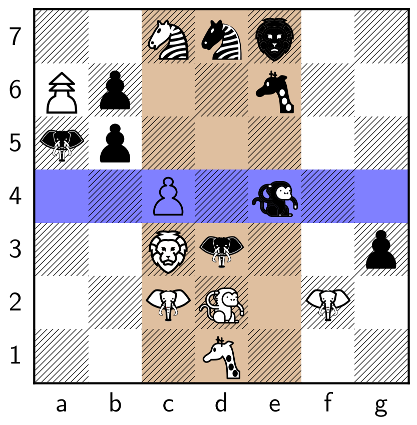
(b) `7/7/2l4/5P1/1PE2p1/2L4/7`

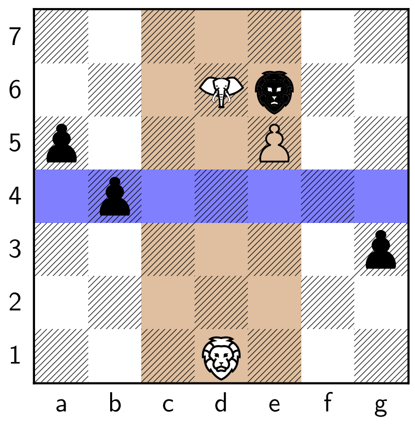


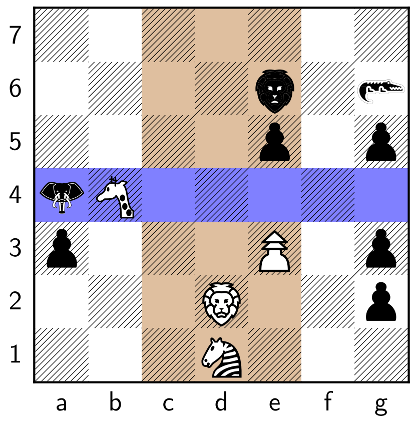
(a) `4Z2/gc1e3/3l3/4E2/S6/4L1m/e1E4`

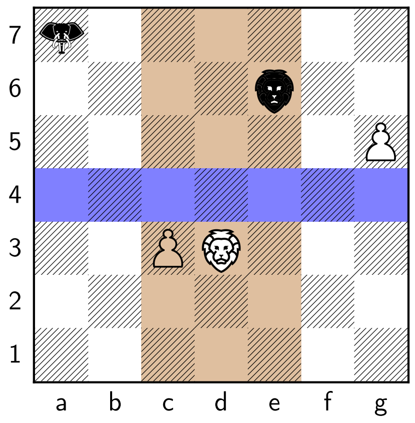


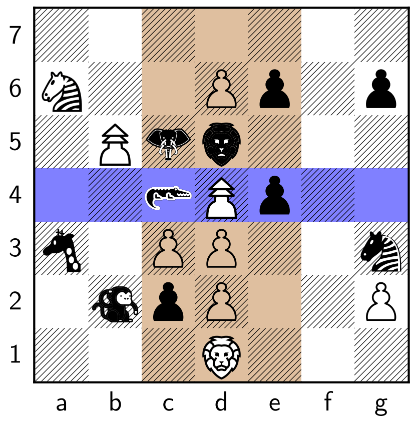
(b) `7/3l2P/3PP1e/p6/2L4/4E2/7`

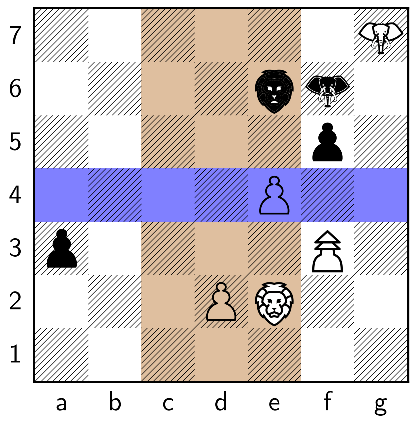

(a) 2Zzl2/Sp2g2/ep5/2P1m2/2Le2p/2EM1E1/3G3

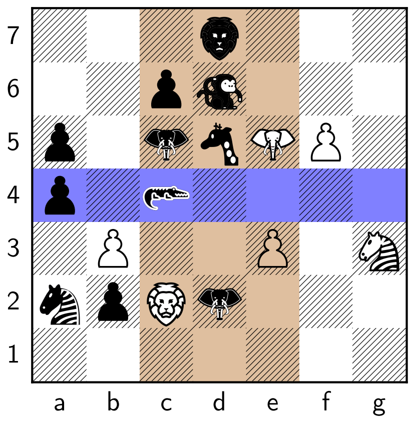

(b) 7/3El2/p3P2/1p5/6p/7/3L3


(a) 7/4l1c/4p1p/eG5/p3S1p/3L2p/3Z3

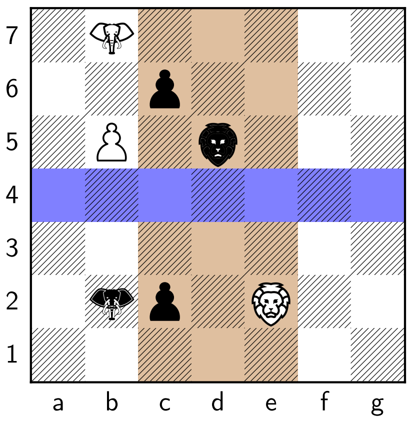

(b) e6/4l2/6P/7/2PL3/7/7


(a) 7/Z2Pp1p/1Sel3/2cSp2/g1PP2z/1mpP2P/3L3

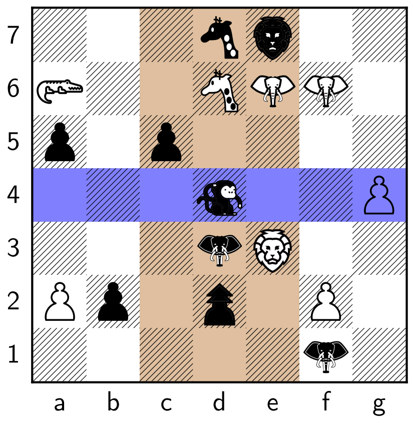

(b) 6E/4le1/5p1/4P2/p4S1/3PL2/7



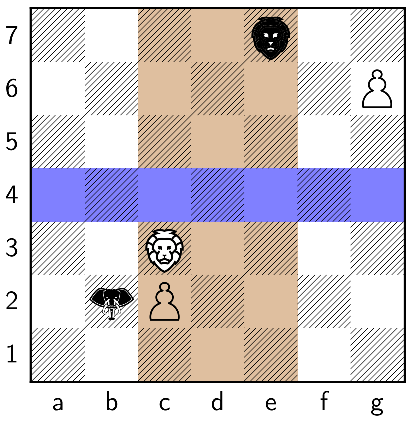
(a) `3l3/2pm3/p1egEP1/p1c4/1P2P1Z/zpLe3/7`



(b) `1E5/2p4/1P1l3/7/7/1ep1L2/7`



(a) `3gl2/C2GEE1/p1p4/3m2P/3eL2/Pp1s1P1/5e1`



(b) `4l2/6P/7/7/2L4/1eP4/7`