UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG
School of Computer Science & Applied Mathematics

COMS4033A/7044A Assignment
Next State Prediction

1 Introduction

Now that we have a way of representing the state and executing actions, the next step is to predict future possible states, given that the opponent has made an unseen move.

Recall that this version of chess is slightly different to standard chess. In particular, it is possible that at the end of a move, the player's king is under attack. This is illegal in standard chess, but legal in this version of chess, since the aim is to capture the opposing king. As a result, more valid moves are available to an agent than would otherwise be in standard chess.

Fortunately, most of these moves already have a name—pseudolegal moves—and are supported by the `python-chess` package. However, in Reconnaissance Blind Chess, there are two additional kinds of moves not included in the set of pseudolegal moves. These are:

1. The null move, where no piece is moved. This move is also provided by the `python-chess` package.
2. Specific castling moves. In particular, regular chess does not allow a castling move if there is an opponent's piece attacking any square between the king and rook. This is a much harder condition to check, but fortunately the `reconchess` package provides a number of utility functions in `utilities.py` to make this easier. For reference, sample code is given below to show how these castling moves can be generated:

```
for move in without_opponent_pieces(board).generate_castling_moves():
    if not is_illegal_castle(board, move):
            # this would be a valid castling move in RBC
            print(move)
```

2 Next Move Prediction

Given a position encoded by a FEN string, your task is to generate all possible next moves that can be played under the current rules. Write a Python program that accepts a FEN string as input and outputs the next possible moves (note that the null move is encoded as 0000). The output should be arranged in alphabetical order.

Sample Input #1

```
8/5k2/8/8/8/p1p1p2n/P1P1P3/RB2K2R w K - 12 45
```

Sample Output #1

```
0000
e1d1
e1d2
e1f1
e1f2
e1g1
h1f1
h1g1
h1h2
h1h3
```

Sample Input #2

```
8/8/8/8/7q/p2p1p1k/P2P1P2/Rn2K2R w KQ - 23 30
```

Sample Output #2

```
0000
a1b1
e1d1
e1e2
e1f1
e1g1
h1f1
h1g1
h1h2
h1h3
```

3 Next State Prediction

Given a position encoded by a FEN string, your next task is to generate all possible next positions that could arise. These positions would be the outcome of all the legal moves (as generated in the previous task). Write a Python program that accepts a FEN string as input and outputs FEN strings corresponding to the next possible positions. The output should be arranged in alphabetical order.

Sample Input #1

```
8/8/8/8/k7/8/7K/3B4 w - - 48 32
```

Sample Output #1

```
8/8/8/7B/k7/8/7K/8 b - - 49 32
8/8/8/8/B7/8/7K/8 b - - 0 32
8/8/8/8/k5B1/8/7K/8 b - - 49 32
8/8/8/8/k7/1B6/7K/8 b - - 49 32
8/8/8/8/k7/5B2/7K/8 b - - 49 32
8/8/8/8/k7/6K1/8/3B4 b - - 49 32
8/8/8/8/k7/7K/8/3B4 b - - 49 32
8/8/8/8/k7/8/2B4K/8 b - - 49 32
8/8/8/8/k7/8/4B2K/8 b - - 49 32
8/8/8/8/k7/8/6K1/3B4 b - - 49 32
8/8/8/8/k7/8/7K/3B4 b - - 49 32
8/8/8/8/k7/8/8/3B2K1 b - - 49 32
8/8/8/8/k7/8/8/3B3K b - - 49 32
```

Sample Input #2

```
k7/p2p1p2/P2P1P2/8/8/8/8/7K b - - 23 30
```

Sample Output #2

```
1k6/p2p1p2/P2P1P2/8/8/8/8/7K w - - 24 31
8/pk1p1p2/P2P1P2/8/8/8/8/7K w - - 24 31
k7/p2p1p2/P2P1P2/8/8/8/8/7K w - - 24 31
```

4 Next State Prediction with Captures

The previous submission assumed no knowledge— our opponent could have played any move. However, Reconnaissance Blind Chess provides the player with various forms of information that can narrow down what moves may have been played. One of the most straightforward indications is when the opponent captures one of our pieces. In this case, we know that it is only moves that have made a capture on the relevant square that could have been played—no other moves would result in that information being sent to us.

Given a position (as a FEN string) and a square on which a capture took place (where squares are indicated by letters and numbers as in the diagram below), generate the set of next possible states in alphabetical order. The next possible states are all outcomes of moves that make a capture on the given square.

	a	b	c	d	e	f	g	h	
8	a8	b8	c8	d8	e8	f8	g8	h8	8
7	a7	b7	c7	d7	e7	f7	g7	h7	7
6	a6	b6	c6	d6	e6	f6	g6	h6	6
5	a5	b5	c5	d5	e5	f5	g5	h5	5
4	a4	b4	c4	d4	e4	f4	g4	h4	4
3	a3	b3	c3	d3	e3	f3	g3	h3	3
2	a2	b2	c2	d2	e2	f2	g2	h2	2
1	a1	b1	c1	d1	e1	f1	g1	h1	1
	a	b	c	d	e	f	g	h	

Sample Input #1

```
k1n1n3/p2p1p2/P2P1P2/8/8/8/8/7K b - - 23 30
d6
```

Sample Output #1

```
k1n5/p2p1p2/P2n1P2/8/8/8/8/7K w - - 0 31
k3n3/p2p1p2/P2n1P2/8/8/8/8/7K w - - 0 31
```

Sample Input #2

```
r1bqk2r/pppp1ppp/2n2n2/4p3/1b2P3/1P3N2/PBPP1PPP/RN1QKB1R w KQkq - 0 5
e5
```

Sample Output #2

```
r1bqk2r/pppp1ppp/2n2n2/4B3/1b2P3/1P3N2/P1PP1PPP/RN1QKB1R b KQkq - 0 5
r1bqk2r/pppp1ppp/2n2n2/4N3/1b2P3/1P6/PBPP1PPP/RN1QKB1R b KQkq - 0 5
```

5 Next State Prediction with Sensing

The main feature of Reconnaissance Blind Chess is the ability to sense a 3×3 window on the board. Given a list of potential states, we can use this window to rule out those that are inconsistent with the window observation.

Write a Python program that takes in a set of potential states and a window, and outputs those states consistent with the window observation. States are encoded as FEN strings as usual, while the window is encoded in the following format:

```
<square>:<piece>;<square>:<piece>;<square>:<piece>;<square>:<piece>;<square>:<piece>;
<square>:<piece>;<square>:<piece>;<square>:<piece>;<square>:<piece>
```

In the above, each square is a location on the board (as in the above diagram) and the piece at that location. White pieces are indicated using upper case characters (`K,Q,R,B,N,P`), and black pieces lower case characters (`k,q,r,b,n,p`). These represent the king, queen, rook, bishop, knight and pawn respectively. If no piece is present, then this is indicated by the character `?`

The first line of input is an integer N, indicating that there are N states under consideration. This is followed by N FEN strings. The next line is the window description.

Output the list of potential states that are consistent with the window observation in alphabetical order

Sample Input #1

```
3
1k6/1ppn4/8/8/8/1P1P4/PN3P2/2K5 w - - 0 32
1k6/1ppnP3/8/8/8/1P1P4/PN3P2/2K5 w - - 0 32
1k6/1ppn1p2/8/8/8/1P1P4/PN3P2/2K5 w - - 0 32
c8:?;d8:?;e8:?;c7:p;d7:n;e7:?;c6:?;d6:?;e6:?
```

Sample Output #1

```
1k6/1ppn1p2/8/8/8/1P1P4/PN3P2/2K5 w - - 0 32
1k6/1ppn4/8/8/8/1P1P4/PN3P2/2K5 w - - 0 32
```