RAICHU

Raichu is a popular childhood game played on an n × n grid (where n ≥ 8 is an even number) with three kinds of pieces (Pichus, Pikachus, and Raichus) of two different colors (black and white). Initially the board starts empty, except for a row of white Pikachus on the second row of the board, a row of white Pichus on the third row of the board, and a row of black Pichus on row n − 2 and a row of black Pikachus on row n − 1.

	1	2	3	4	5	6	7	8
1	.	.	.	.	.	.	.	.
2	W	.	W	.	W	.	W	.
3	.	w	.	w	.	w	.	w
4	.	.	.	.	.	.	.	.
5	.	.	.	.	.	.	.	.
6	b	.	b	.	b	.	b	.
7	.	B	.	B	.	B	.	B
8	.	.	.	.	.	.	.	.

Two players alternate turns, with White going first. In any given turn, a player can choose a single piece of their color and move it according to the rules of that piece. A Pichu can move in one of two ways:

- one square forward diagonally, if that square is empty. - "jump" over a single Pichu of the opposite color by moving two squares forward diagonally, if that square is empty. The jumped piece is removed from the board as soon as it is jumped.

For example, for the highlighted Pichu in the following board in figure 1, there are two possible moves, shown in the figures 2 and 3:

	1	2	3	4	5	6	7	8
1	.	.	.	.	.	.	.	.
2	W	.	W	.	W	.	W	.
3	.	w	.	w	.	w	.	w
4	.	.	b	.	.	.	.	.
5	.	.	.	.	.	.	.	.
6	b	.	b	.	.	.	b	.
7	.	B	.	B	.	B	.	B
8	.	.	.	.	.	.	.	.

Figure 1: Initial board

Figure 2: Possible move #1

Figure 3: Possible move #2

A Pikachu can move in one of two ways:

- 1 or 2 squares either forward, left, or right (but not diagonally) to an empty square, as long as all squares in between are also empty.
- "jump" over a single Pichu/Pikachu of the opposite color by moving 2 or 3 squares forward, left or right (not diagonally), as long as all of the squares between the Pikachu's start position and jumped piece are empty and all the squares between the jumped piece and the ending position are empty. The jumped piece is removed as soon as it is jumped.

For example, for the highlighted Pikachu in the following board in figure 4, here are some (not all) possible moves:

Figure 4: Initial board

Figure 5: Possible move #1

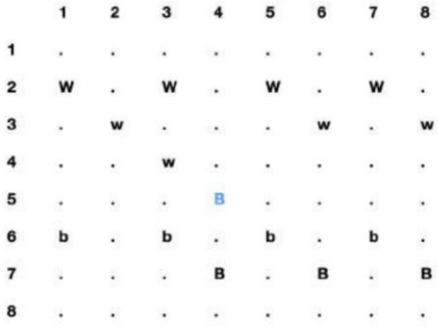

Figure 6: Possible move #2

Figure 7: Possible move #3

A Raichu is created when a Pichu or Pikachu reaches the opposite side of the board (i.e. when a Black Pichu or Pikachu reaches row 1 or a white Pichu or Pikachu reaches row n). When this happens, the Pichu or Pikachu is removed from the board and substituted with a Raichu.

Raichus can move as follows:

- any number of squares forward/backward, left, right or diagonally, to an empty square, as long as all squares in between are also empty.
- "jump" over a single Pichu/Pikachu/Raichu of the opposite color and landing any number of squares forward/backward, left, right or diagonally, as long as all of the squares between the Raichu's start position and jumped piece are empty and all the squares between the jumped piece and the ending position are empty. The jumped piece is removed as soon as it is jumped.

For example, some (not all) possible moves of the highlighted white Raichu are:

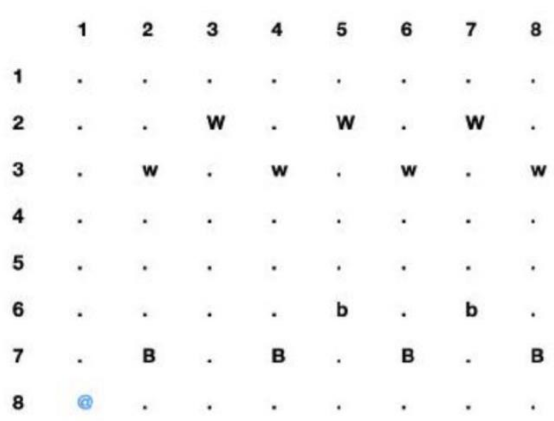

Figure 8: Initial board

Figure 9: Possible move #1

Figure 10: Possible move #2

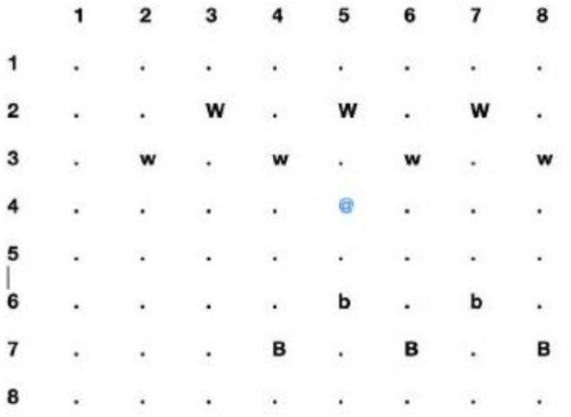

Figure 11: Possible move #3

Your task is to write a Python program that plays Raichu well. Your program should accept a command line argument that gives the current state of the board as a string of .'s, w's, W's, b's, B's, @'s, and $'s, which indicate which squares have no piece, a white Pichu, a white Pikachu, a black Pichu, a black Pikachu, a white Raichu and a black Raichu respectively, in row-major order. For example, if n = 8, then the encoding of the start state of the game would be:

.....W.W.W.W..w.w.w.w.....b.b.b.b..B.B.B.B.....

More precisely, your program will be called with four command line parameters: (1) the value of n, (2) the current player (w or b), (3) the state of the board, encoded as above, and (4) a time limit in seconds. Your program should then decide a recommended single move for the given player with the given current board state, and display the new state of the board after making that move. Displaying multiple lines of output is fine as long as the last line has the recommended board state. The time limit is the amount of time that your program should expect to have to make its decision; our testing code will kill your program at that point, and will use whichever was the last move your program recommended. For example, a sample run of your program might look like:

[hrupchan@silo]$ python3 raichu.py 8
w '.....W.W.W.W..w.w.w.w.....b.b.b.b..B.B.B.B.....' 10
Searching for best move for w from board state:
.....
W.W.W.W.
.w.w.w.w
.....
.....
b.b.b.b.
.B.B.B.B
.....
Here's what I decided:
.....W.W.W..w.w.w.wW.....b.b.b.b..B.B.B.B.....