Contents

1 Function specifications

The functions must be implemented **exactly as per the specifications below**. If they do not meet the specifications, you will not have a good time. **Your functions do not meet the specifications** if:

- you modify the function inputs in any way such that the inputs and outputs are not exactly as described; or
- your function relies on other variables that are not in its input parameters; or
- your function relies on additional input from the user apart from its input parameters (i.e. any `input()` calls).

You may presume (reasonably) that the input is exactly as described, and that only sensible inputs are given. Input validation is not necessary, but is easy enough to implement.

The gameplay rules are not described here.

1.1 Representing the game board

The game board is represented as a list of n^2 integers, consisting of only the values 0, 1, and 2.

- Zeros (`0`) in the list represent a blank tile, i.e. an unplayed tile.
- Ones (`1`) in the list represent a tile for player 1.
- Twos (`2`) in the list represent a tile for player 2.

The position of each integer in the list represents a tile in a fixed location in the board. In a board of size n, the first n entries represent the first row, the next n entries represent the second row, and so on, until the n-th set of n entries, representing the n-th row.

For example, in a board of size 3 there are 9 integers, the first 3 represent the first row, the next set of 3 represent the second row, and the 3rd set of 3 represent the 3rd row. The board `[1, 2, 0, 0, 1, 1, 2, 0, 1]` represents the board state below, with player 1's tiles (arbitrarily) represented with an O, and player 2's tiles (arbitrarily) represented with an X.

O	X	
	O	O
X		O

1.2 `check_move(board, turn, index, push_from)`

Inputs:

- `board` - a list of size n^2, consisting of only the values 0, 1, and 2, representing the state of the board.

- `turn` - an integer, either 1 or 2, representing whose turn is being played.
- `index` - an integer between 0 and $n^2 - 1$, inclusive of both ends, representing the position at which the player's move was made.
- `push_from` - a string, one of `'T'`, `'B'`, `'L'`, or `'R'`, representing the direction in which the tile is pushed from according to the gameplay rules.

Outputs:

A boolean, `True` or `False`, depending on whether the move (`index` and `push_from`) played by player `turn` with a board state `board` was a legal move or not, according to the gameplay rules.

1.3 apply_move(board, turn, index, push_from)

Inputs:

- `board` - a list of size n^2, consisting of only the values 0, 1, and 2, representing the state of the board.
- `turn` - an integer, either 1 or 2, representing whose turn is being played.
- `index` - an integer between 0 and $n^2 - 1$, inclusive of both ends, representing the position at which the player's move was made.
- `push_from` - a string, one of `'T'`, `'B'`, `'L'`, or `'R'`, representing the direction in which the tile is pushed from according to the gameplay rules.

The move being played, `index` and `push_from`, are always legal moves according to the gameplay rules.

Outputs:

A list of size n^2, consisting of only the values 0, 1, and 2, representing the state of the board, after the move (`index` and `push_from`) are played by player `turn` with board state `board`.

1.4 check_victory(board, who_played)

Inputs:

- `board` - a list of size n^2, consisting of only the values 0, 1, and 2, representing the state of the board.
- `who_played` - an integer, either 1 or 2, representing whose turn was played leading to the current state of the board.

Outputs:

An integer, `0` if there is no winner, `1` if player 1 wins, `2` if player 2 wins, according to the gameplay rules.

1.5 computer_move(board, turn, level)

Inputs:

- `board` - a list of size n^2, consisting of only the values 0, 1, and 2, representing the state of the board.
- `turn` - an integer, either 1 or 2, representing whose turn was played.

Outputs:

A tuple of two items, ordered as such:

1. An integer between 0 and $n^2 - 1$, inclusive of both ends, representing the position at which the computer player's move is to be played.
2. A string, one of `'T'`, `'B'`, `'L'`, or `'R'`, representing the direction in which the tile computer player's move should be pushed from according to the gameplay rules.

1.6 display_board(board)

Inputs:

- `board` - a list of size n^2, consisting of only the values 0, 1, and 2, representing the state of the board.

Outputs:

The function **does not return anything**. It displays the given board to the user.

1.7 menu()

Inputs: none.

Outputs: none.

Its purpose is meant to act as the main gameplay loop, taking user input and directing how the game flows.