APS 105 — Computer Fundamentals

Project: Reversi Game (Part 2)

Complementary Documentation for makeMove function

Winter 2021

The function "makeMove" is where the computer makes its move. In part1 of lab8, "makeMove" implement a move for computer based on the simplest way of "greedy" method of scoring, as described in the handout. However, in part2 "makeMove" function **should be modified** based on your "smarter strategy".

This documentation will explain the implementation detail of the function which is not well documented in the lab handout:

```
int makeMove(const char board[26][26], int n, char turn, int *row, int *col);
```

Parameter:

This function takes five parameters, the board, n (dimension of the board), turn (who should take the turn, Black or White?), and two pointers to two integer value.

Expected goal of this function:

Given the board, dimension of the board, and the color of next move, this function should calculate a best move for that color. You are NOT supposed to update the board within this function (the parameter has a const keyword indicating that you shouldn't modify anything stored in that array). The location of the best move should return using the two pointers (`row` and `col`) to two integer values. Notice that the function should return an integer type. It is up to you what you want to return. If you think you don't have any more information to return from this function, you can simply return 0.

Example program of how this function works:

```
char board[26][26];
int n;
char turn;
int row, col;

/*
 * board, n, and turn are set here
 * …
 */
int ret = makeMove(board, n, turn, &row, &col);

// update the board here…
```

Example program memory status during runtime:

Example from lab handout:

```
Enter the board dimension: 4
Computer plays (B/W) : W
abcd
a UUUU
b UWBU
c UBWU
d UUUU
Enter move for colour B (RowCol): ba
abcd
a UUUU
b BBBU
c UBWU
d UUUU
Computer places W at aa. /* what happened here? */
```

Before makeMove function is called:

board	- integer array:	[board configuration] UUUU BBBU UBWU UUUU
n	- integer:	4
turn	- integer:	'W'
row	- integer ptr:	points to some valid address, but the content is irrelevant
col	- integer ptr:	points to some valid address, but the content is irrelevant

Computer's turn; makeMove is called as follows:

board	- integer array:	[board configuration] UUUU BBBU UBWU UUUU
n	- integer:	local variable, memory gets recycled.
turn	- integer:	local variable, memory gets recycled.
row	- integer ptr:	points to some valid address, the content gets updated to 0 ('a')
col	- integer ptr:	points to some valid address, the content gets updated to 0 ('a')

Note:

1. **ALL your logics for your AI (lab8 part2) should remain in the scope of this function**. Otherwise, your code will behave unexpectedly when other students want to compete against your AI online through the online interactive platform.
2. **NO** global variable is allowed. Otherwise, your AI may return unexcepted result.

Implementation for part1 and part2:

In part1 of this lab, you should follow the lab handout. Therefore, you should output the exact same move as described in the lab handout.

In part 2 of this lab, you need to implement your own AI. You can use any algorithm you like as long as you follow the requirements stated in this documentation, lab handout and requirement from lab7.