Project 2 – Reversi

Assignment: Project 2 – Reversi
Due Date:
Design Document: Friday, April 12th, 2019 by 11:59:59 PM
Project: Friday, April 19th, 2019 by 11:59:59 PM
Value: 80 points

Collaboration: For Project 2, **collaboration is not allowed** – you must work individually. You may still come to office hours for help, but you may not work with any other CMSC 201 students.

Make sure that you have a complete file header comment at the top of each file, and that all of the information is correctly filled out.

```
# File:     FILENAME.py
# Author:   YOUR NAME
# Date:     THE DATE
# Section:  YOUR DISCUSSION SECTION NUMBER
# E-mail:   YOUR_EMAIL@umbc.edu
# Description:
#    DESCRIPTION OF WHAT THE PROGRAM DOES
```

Change Log

4/8/19:

- Formatting: Added description of supplied printBoard() with instructions on where to find it.
- Input validation: Added an entry requiring that a row and column be found from user input
- Change to sample output: some small changes to the accompanying sample output

For Project 2 you will have to turn in a "design document" in addition to the actual code. The design document is intended to help you practice deliberate construction of your program and how it will work, rather than coding as you go along, or starting without a plan.

Instructions

For this project, you will be creating a single program, but one that is bigger in size and complexity than any individual homework problem. This assignment will focus on manipulating lists, calling functions, and handling mutability appropriately.

The design for Project 2 is entirely up to you – suggestions are provided within the project description, but you are not required to use them.

At the end, your Project 2 file must run without any errors.
It must also be called proj2.py (case sensitive).

Additional Instructions – Creating the proj2 Directory

During the semester, you'll want to keep your different Python programs organized, organizing them in appropriately named folders (also known as directories).

You should create a directory in which to store your Project 2 files. We recommend calling it **`proj2`**, and creating it inside a newly-created directory called **`Projects`** inside the **`201`** directory.

If you need help on how to do this, refer back to the detailed instructions in Homework 1.

Objective

Project 2 is designed to give you practice with two-dimensional lists, mutability, and creating and calling functions. You'll need to use practically everything you've learned so far, and will need to do some serious thinking about how all of the pieces you need to create should fit together.

Task

For this project, you will be coding a very simplified game of Reversi, in which there is only one player, and the opponent is a computer player that follows a simple naive strategy. The game is played on an 8x8 board.

Coding Standards

Prior to this assignment, **you should be familiar with the entirety of the Coding Standards**, available on Blackboard under "Assignments" and linked on the course website at the top of the "Assignments" page.

You should be commenting your code, and using constants in your code (not magic numbers or strings).
Any strings with a meaning should be constants!
Any numbers other than 0 or 1 are magic numbers!

You will **lose major points** if you do not follow the 201 coding standards.

If you have questions about commenting, whitespace, or any other coding standards, please come to office hours.

Additional Specifications

For this assignment, you must create and call **at least eight individual functions**, not including **`main()`**. All other design decisions are up to you.

You may *not* import any libraries or use any library functions! Doing so will lose you a *very* large number of points.

Input Validation

For this project, we will require that you validate input from the user. You can assume that the user will enter the right type of input, but not that they will enter a correct value. In other words, a user will always give an integer when you expect one, but it may be a negative or otherwise invalid value.

You will need to validate the following things:

- Getting the user's choice for each move, both row and column
  - Rows and columns must both be between 0 and 7, inclusive
- The player's move must be a LEGAL Reversi move for the current board
- The input must contain both a row and column

Formatting

When printed, your board must appear ***exactly*** like the one shown in the sample output, including the row and column headings, and the dividing lines.

You can look at the sample output to see what a board should look like, but a description has also been provided below with exact character counts and descriptions.

The board shown in the sample output uses underscores for empty spaces, and vertical bars (*i.e.*, pipes) for the vertical dividing lines. Each square is one character wide, and contains an uppercase X for a player move, an O for a computer move, and an underscore for an empty space. The row and column headings follow this same format, but with a number for the row or column index. The final column label is not followed by a pipe.

To this end, we have provided a **`printBoard()`** function for you to use, which will print the board exactly as shown in the sample output. You are not required to use this function, but we recommend that you do, as it has been tested and works correctly. You are also welcome to modify it to suit your needs.

The code for the function is available as as a separate file under "Assignments" on Blackboard, and is called "**`printBoard.txt`**".

You can also get it from Dr. Gibson's pub folder using the following command:

```
cp /afs/umbc.edu/users/k/k/k38/pub/cs201/printBoard.txt .
```

Details

For this project, you will be coding a simplified game of Reversi. The board for the game is an 8x8 grid. Players take turns placing X's and O's into blank spaces on the board.

The Setup

The board is set up with four pieces already played in the middle:

```
Current board state:
_|0|1|2|3|4|5|6|7
0|_|_|_|_|_|_|_|_|
1|_|_|_|_|_|_|_|_|
2|_|_|_|_|_|_|_|_|
3|_|_|_|X|O|_|_|_|
4|_|_|_|O|X|_|_|_|
5|_|_|_|_|_|_|_|_|
6|_|_|_|_|_|_|_|_|
7|_|_|_|_|_|_|_|_|
```

BE SURE THAT YOU DON'T GET THE X's AND O's BACKWARDS!

Taking a move

Before the player's turn, you will display all of their valid moves, and then prompt them for a move. They will give the move as two integers with a space in between. If the integers are NOT valid moves, you will reprompt them.

A valid Reversi move for the player (placing pieces represented by X) is defined as:

- Horizontally or vertically adjacent to an O with the following properties
  - The adjacent O must be in a line O's of length at least one
  - The end of the line of O's must terminate in an X

More simply put, an X must be placed so that O's are "sandwiched" between the new X and an existing one. An example board position:

```
Current board state:
_|0|1|2|3|4|5|6|7
0|_|_|_|_|_|_|_|_|
1|_|_|_|_|_|_|_|_|
2|_|_|_|O|_|_|_|_|
3|_|_|_|O|X|X|_|_|
4|_|_|_|O|X|_|_|_|
5|_|_|_|_|_|_|_|_|
6|_|_|_|_|_|_|_|_|
7|_|_|_|_|_|_|_|_|
Valid moves are: [[3, 2], [4, 2]]
```

Seasoned Reversi players will notice that we have not mentioned diagonals. ***YOU MUST IGNORE DIAGONAL PLAYS FOR THIS PROJECT***. Projects that include diagonal plays will lose a **major amount** of points.

After the move is placed, all of the "sandwiched pieces" for the opponent are "flipped" to the control of the player that just moved. More precisely:

> If a piece is played, it must be next to one or more sets of enemy pieces that are in a line between the newly played piece and another friendly piece. All of those enemy pieces are "flipped" to be controlled by the player who just played a piece.

IMPORTANT NOTE: ***ALL*** pieces that should be flipped, must be flipped, in all directions (up, down, left, right) from the placed piece.

Example (flipped pieces in red: [3, 2], [2, 3])

```
_|0|1|2|3|4|5|6|7                        _|0|1|2|3|4|5|6|7
0|_|_|_|_|_|_|_|_|                       0|_|_|_|_|_|_|_|_|
1|_|_|_|_|X|_|_|_|                       1|_|_|_|_|X|_|_|_|
2|_|_|_|O|X|O|_|_|   -User plays 2 2->   2|_|_|X|X|X|O|_|_|
3|_|O|O|O|X|O|O|_|                       3|_|O|X|O|X|O|O|_|
4|_|O|X|X|X|_|_|_|                       4|_|O|X|X|X|_|_|_|
5|_|O|O|_|_|_|_|_|                       5|_|O|O|_|_|_|_|_|
6|_|_|_|_|_|_|_|_|                       6|_|_|_|_|_|_|_|_|
7|_|_|_|_|_|_|_|_|                       7|_|_|_|_|_|_|_|_|
```

Artificial Unintelligence

The computer player will always make the legal move closest to the top. If there are two or more legal moves in the same row, the computer prefers the move to the left. Your computer player **MUST** play this way in order to get full credit for the assignment.

Why so dumb, you may wonder? Having the computer play moves that follow a simple pattern let us give you more useful sample outputs than if our computer played random moves, or moves based on an algorithm of any level of sophistication.

If you love Reversi, and want to make a smarter computer player, do it AFTER you submit your assignment.

Turn flow

A turn is comprised of the following:

1. The board is displayed.
2. The user is prompted for a move
3. The user inputs the move

4. The computer's move is printed to the player (but NOT the board).

Steps 1-4 are repeated until the game ends. Do not print the board state before the computer plays.[1]

End of the Game

The game ends when the player whose turn it is has no legal move to make. This includes the computer player! Again, this deviates from standard Reversi rules, so be sure that you implement accordingly to this project description.

Hints and Advice

It would be a good idea to:

- Store your board in a 2D list (make sure you don't mix up column and row!)
- Remember that lists are passed by reference, and that any in-place changes to a list made within a function remain when control is returned to the calling function
- Use constants for the different characters used on the board, as well as any other numbers or strings you deem necessary
- Have a function called **`printBoard()`** that takes in the current board as a parameter and prints out the board's contents
- When printing the board, remember to make use of either concatenation or the **`end=""`** parameter in the **`print()`** function, to control when line breaks and spaces are printed to the screen.

TIP: This would be a very good time to use incremental development! Incremental development is when you are only working on a small piece of the code at a time, and testing that the piece of code works before moving on to the next piece. This makes it a lot easier to fix any mistakes.

[1] You are welcome to print the board after the player makes their move as a part of your debugging, but remove it before submitting. As a "final product", having a lot of boards displayed can be confusing to the player.

Project

The project is worth a total of 80 points. Of those points 10 will be based on your design document, 10 will be based on following the coding standards, and the other 60 will be based on the functionality and completeness of your project.

Design Document

The design document will ensure that you begin seriously thinking about your project early on. This will not only give you important experience doing design work, but will help you gauge the number of hours you'll need to set aside to be able to complete the project. **Your design document must be called design2.txt.**

For Project 2, you are creating the design entirely on your own.
You **may NOT work with another student** to "brainstorm" a solution or discuss any general approaches or requirements. If you need assistance with the design document, come to office hours.

Your design document must have four separate parts:

1. A file header, similar to those for your assignments
2. Constants
   a. A list of all the constants your program will need, including a short comment describing what each "group" of constants is for (*e.g.,* menu options, meaning of indexes, etc.)
3. Function headers
   a. A complete function header comment for each function you plan to create, including the name, description, parameters, and return value.
4. Pseudocode for main()
   a. A brief but descriptive breakdown of the steps your main() function will take to completely solve the problem; note function calls under the relevant comment (if applicable)

Your design can follow the same general format as the design for Project 1.

Your **`design2.txt`** file will be compared to the **`proj2.py`** file that you submit. Minor changes to the design are allowed. A minor change might be the addition of another function, or a small change to **`main()`**.

Major changes between the design and your project will lose you points. This would indicate that you didn't give sufficient thought to your design.
(If your submitted design doesn't work, it is generally better to lose the points on the design, and to have a functional program, rather than turning in a broken program that follows the design. The decision is ultimately up to you.)

To submit your design document, use

```
linux1[4]% submit cs201 PROJ2_DESIGN design2.txt
Submitting design2.txt...OK
linux1[5]%
```

Sample Output

The sample output is available as a separate file under "Assignments" on Blackboard, and is called "sample2.txt".

(Yours does not have to match the sample output exactly, but it should be similar.)

Submitting

Once your **`proj2.py`** or **`design2.txt`** file is complete, it is time to turn it in with the **`submit`** command. (You may also turn the design or project in multiple times, as you reach new milestones or complete each piece. To do so, run **`submit`** as normal.)

To submit your design file (which is due Friday, April 12th, 2019 by 11:59:59 PM), use the command:

```
linux1[4]% submit cs201 PROJ2_DESIGN design2.txt
Submitting design2.txt...OK
linux1[5]%
```

To submit your project file (which is due Friday, April 19th, 2019 by 11:59:59 PM), use the command:

```
linux1[4]% submit cs201 PROJ2 proj2.py
Submitting proj2.py...OK
linux1[5]%
```

If you don't get a confirmation like the one above, check that you have not made any typos or errors in the command.

You can check that your homework was submitted by following the directions in Homework 0. Double-check that you submitted your homework correctly, since **an empty file will result in a grade of zero for this assignment.**