CMSC 201 Fall 2018
Project 2 – Connect Four

Assignment: Project 2 – Connect Four
Due Date:
Design Document: Friday, November 2nd, 2018 by 8:59:59 PM
Project: Friday, November 9th, 2018 by 8:59:59 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 your 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
```

Project 2 is the first assignment where we won't be telling you exactly what to do! You will get the chance to make your own decisions about how you want your program to handle things, what its functions should be called, and how you want to go about designing it.

Project 2 is ***substantially longer*** than any of the single homework assignments you've completed so far, so make sure to take the time to plan ahead, and don't do any "cowboy" coding!

Instructions

For this assignment, you'll need to follow the class coding standards, a set of rules designed to make your code clear and readable. The class coding standards are on the website, linked 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 numbers other than 0 or 1 are magic numbers!

Adhere to the coding standards by including function header comments for each of the functions (other than main). Follow the instructions and example provided in the coding standards document when creating your function header comments. Failing to include function header comments will lose you points.

Re-read the coding standards!

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

NOTE: Your filename for this project must be **`proj2.py`**

NOTE: You must use **`main()`** in your file.

Details

For this project, you are going to be building the classic game Connect Four. If you are not familiar with the game, the Wikipedia page has more details (https://en.wikipedia.org/wiki/Connect_Four). In our game, Player 1 is an “x” and Player 2 is an “o”. If any player gets four of their pieces in a row (horizontally, vertically, or diagonally), they win. When the board fills up completely and neither player has won, the game ends in a draw.

The program should start by prompting the user for the desired size of their board: both row and column should be gotten from the user (in that order). Both row and column must be at least 5; if they enter a smaller number, the program must prompt the user again.

Users then choose a column to put their piece in; the piece will be placed at the “bottom” of the column. **Column numbering must start at 1, not zero!** If the user chooses an invalid column or a column that is already full, the program must prompt the user again.

After a game ends (in a draw or in a win), the user should be asked if they want to play again. If they choose to play again, they should be asked for the dimensions of the next game’s board (row and column).

The user can exit the game by choosing “n” after a game has ended, or by hitting CTRL+C at any point in time.

More Details

Some additional requirements:

1. The game will played by two human players, or by one human player and one computer player.
2. Player 1 always goes first when a new game is started. If playing against the computer, then the computer acts as Player 2 (they go second).
3. You must show the updated board between each player's turn.
4. You must always tell the user their options (range of numbers, y/n) when asking them for a choice (see sample output).
5. The game must output a message when a draw happens.
6. The game must output a message when one player wins (and must state which player won).
7. If the user chooses to play another game after one ends, the program must reprompt for new row and column sizes, and use them in the new game.

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:

- When asked to enter the number of rows and columns, the user might enter any whole number. If the number is less than 5, they should be prompted for a new number.
- When asked a yes/no question, the user might enter anything. You should only accept lowercase "y" (for yes) or lowercase "n" (for no). If they enter anything else, they should be prompted again.
- When asked to choose a column to place their piece in, the user might enter any whole number. They will not enter any other strings. If they choose an invalid integer (a column that does not exist, such as -1 or 99), they should be prompted again.

Sample Output

The sample output for this project is long enough that we have made it available as a separate file. You can find it on Blackboard. It contains examples of input validation, winning, drawing, different board sizes, etc.

Your board should be displayed exactly as shown in the sample output. Blank spots should be represented by underscores, player tokens should be represented by “x” (for Player 1) or “o” (for Player 2), and each spot should be separated by a space when it is printed out.

Play against the computer!

This is part where you can have some fun. You should write a function that allows the user to play against the computer should they choose to do so. The computer doesn't have to be super smart and make the best move possible every time. You can write this function to always play the first open column if you wanted to. However, you are free to write as much logic in here as you want! The one in the sample output just picks a random column each time.

You may not import any libraries for this project except for random like you were shown in Project 1.

Advice and Suggestions

Remember what we've covered so far! Do not try to program the entire game up at once, and don't "cowboy code." You need to break this program down into simple pieces that you code and test one at a time.

HINT: You should represent the board as a two-dimensional list.

Here is one possible way you could tackle this problem:

1. Ignore playing against the computer and checking for win or draw. Leave those until you know everything else works.
2. Get the size of the board from the user and create it.
    a. Be able to print out the board.
3. Use a "temporary" **`main()`** to interact and test each function as you create and write them.
4. Figure out how to "switch" turns from one player to another.
5. Be able to validate the user input for choosing a column.
6. Update the board
    a. Make sure it's the right column and the right player's piece.

Once all of that works, and you've tested it completely, then you can start writing the code to handle draws, wins, and playing against the computer. (Work on them separately! Draw is probably the easiest.)
Pay close attention to the sample output provided when testing your own code.

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 make sure that you begin thinking about your project in a serious way early on in the process. 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.**

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
3. Function headers
    a. A complete function header comment for each function
    b. You do not need to include the function's code in your design
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 **`design2.txt`** file will be compared to the **`proj2.py`** file that you submit. Minor changes to the design are acceptable. 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.)

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 project in multiple times, as you reach new milestones. To do so, run **`submit`** as normal.)

To submit your design file (which is due Friday, November 2nd, 2018 by 8:59:59 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 Friday, November 9th, 2018 by 8:59:59 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 assignment 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.**