Final Project CSC111
200pts Spring 2020

You are tasked with making a terminal version of the game Connect Four. Connect Four is played on a 6 X 7 grid (that's 6 rows and 7 columns). The goal is to place your pieces on the board so that you have four pieces in a row all connected. The first player to get four in a row wins. You can get four in a row horizontally, vertically, or diagonally.

Pieces are placed with each player taking turns. The board stands vertically and pieces are dropped into one of the seven column slots at the top of the board. The piece will fall down to the lowest available empty slot. So for instance, here is a sequence of moves two players might make, represented as a series of board printouts (O represents empty, r represents red pieces, and b represents black pieces:

```
0000000    0000000    0000000    0000000    0000000    0000000    0000000    0000000
0000000    0000000    0000000    0000000    0000000    0000000    0000000    0000000
0000000    0000000    0000000    0000000    0000000    0000000    0000000    0000000
0000000    0000000    000b000    000b000    000b000    000b000    00bb000    00bb000
0000000    000r000    000r000    000r000    000r000    00rr000    00rr000    00rr000
000b000    000b000    000b000    00rb000    00rbb00    00rbb00    00rbb00    r0rbb00
```

The game would end if either player lined up four of their pieces as described above.

If you are not familiar with the game, please watch some YouTube videos of people playing or run through an online version against a CPU to get a sense of its dynamics. If you have any questions about the rules or the way it works, please ask me right away.

Your program should start by asking the name of each of the two human players and welcoming them by name to the game. The first player should go first and should use the black pieces. The second player will go next using the red pieces.

The program will alternate prompting each user to choose their next move. The players will choose a move by inputting a number 1-7 indicating which column they would like to drop their piece down. So for instance, in the first move of the example game, the first player chose 4 at the input screen. The program should continue prompting users to pick a position until either the board is full or one of the players has four in a row.

After each turn, your program should check if the board is full or if the player who just made a move connected four pieces in a row. If either of those cases is true, the program loop should break and a winner should be announced. At that point, the program should ask the users if they would like to play again. If they type "y" or "yes" the program should start again with the same user names, the same player order, and an empty board. If they type "n" or "no" the program should thank them for playing, say good bye and close. Also after each turn, a textual representation of the board should be printed to the screen to inform the players of the updated board state.

Your program should be organized neatly into functions. There should be a function for checking if there is a winner, a function for placing pieces on the board based on a chosen column number, a function for checking if the board is full, a function for converting the multidimensional array representation of the board to a string for printing, etc. It is up to you to choose a functional composition that best suits your design goals. I am certainly willing to provide guidance on this aspect of the program if you need help.

You must earn at least 100pts on this assignment to get full credit. If you opt to add any of the extra credit features to this program, those extra points can be used as extra credit towards your lab grade.

Extra Credit Features:

1. (10pts) Keep track of how many games each player has won while the program is running and print out the final score between the users once they decide to leave.

2. (10pts) Write an AI that can play the game against a human player. Let the user choose at the beginning if the game will be one player or two players. The AI need not be sophisticated, so long as it is capable of choosing moves against the player and your program is capable of using the AI to compete against the player in a single player mode.

3. (20pts) Write your code so that it will work for a board of any size, not just 7 columns by 6 rows. Ask the user at the start of a new game what the dimensions of their board should be.

4.(25pts) Keep track of every board state in a game represented as a string and print the sequence of boards from that game to a file. There should be one file per game, so if players decide to play multiple games during the same program run, a new file should be written to for the second game etc. Files from previous runs of the program should not be overwritten by new games being played, so make sure you have a way of making each filename unique.

5. (35pts) Add a save feature so that if a player types ‘s’ at any point during play the state of the board along with the player names is saved to a filename of the user’s choosing and the program exits. Whenever the program starts, allow the user to choose to start a new game or continue from a saved game. If they choose to continue from a saved game, allow them to type the name of their saved game file and have that save file load the state of the game.