Homework 5

Files to submit: **all files need to compile connectn.out, a Makefile for connectn.out**
Time it took Matthew to Complete**: 50 mins**

- All programs must compile without warnings when using the -Wall and -Werror options
- Submit only the files requested
  - Do **NOT** submit folders or compressed files such as .zip, .rar, .tar, .targz, etc
- Your program must match the output exactly to receive credit.
  - Make sure that all prompts and output match mine exactly.
  - Easiest way to do this is to copy and paste them
- All input will be valid unless stated otherwise
- Print all real numbers to two decimal places unless otherwise stated
- The examples provided in the prompts do not represent all possible input you can receive.
- All inputs in the examples in the prompt are underlined
  - You don't have to make anything underlined it is just there to help you differentiate between what you are supposed to print and what is being given to your program
- If you have questions please post them on Piazza

Restrictions

- No global variables are allowed
- Your main function may only declare variables, call other functions, and assign variables values.
- **Your submission must consist of at least 2 .c files and 1 .h file.**

1. Write a program to implement the game connect-n. Connect-n is like Connect-4 except instead of having the board be a constant 6 X 7 we will allow the user to enter the size of the board they would like to play on. In addition we will also allow the user to choose how many pieces in a row are necessary to win. The game is played as follows. Two players take turns dropping their pieces into a column until either player gets N of their pieces in a row either horizontally, vertically, or diagonally, or until their or no more spaces to play.
    1. Your program must accept 3 command line parameters in this order: number of rows, number of columns, number of pieces in a row to win
        1. If the user does not enter enough arguments or enters too many arguments your program should tell them the proper usage of your program and terminate.
            1. You may find the exit function helpful
            2. The user should be allowed to create an unwinable game
                1. For example a board that is 3 X 3 but requires 4 pieces in a row to win
    2. Your program should not allow the user to make an impossible play but should continue to ask the user for a play until a valid play is entered
        1. Invalid plays consist of plays made outside the board or in to columns that are already full
    3. The token used to represent Player 1 is X
    4. The token used to represent Player 2 is O, a capitol oh and not a zero
    5. After the game is over the winner should be declared or if there is no winner a tie should be declared
    6. You must split your code up into at least 2 files.
        1. I personally had 4 separate c files
    7. You must submit a make file named Makefile that when run compiles your program
    8. The executable created by your make file should be named **connectn.out**
    9. Hints
        1. This is your first “large” program. It took me around 300 lines of code to complete. You will want to break your problem down into many small manageable functions to make the problem easier to deal with
            1. I also recommend testing each function your write as you go along to help you locate your errors early
        2. I had the following functions defined in my solution and they give a pretty good ordering of how you should write you should solve this problem
            1. read_args
            2. create_board, print_board, destroy board
            3. play_game, get_play, play_is_valid
            4. game_over, game_won, row_win, col_win, diag_win, right_diag_win, left_diag_win.

10. Examples

1. ./connectn.out
   1. Not enough arguments entered
   2. Usage connectn.out num_rows num_columns
      number_of_pieces_in_a_row_needed_to_win
2. ./connectn.out 1 2 3 4 5
   1. Too many arguments entered
   2. Usage connectn.out num_rows num_columns
      number_of_pieces_in_a_row_needed_to_win
3. ./connectn.out 3 3 3

```
2 * * *
1 * * *
0 * * *
  0 1 2
Enter a column between 0 and 2 to play in: 0
2 * * *
1 * * *
0 X * *
0 1 2
Enter a column between 0 and 2 to play in: 1
2 * * *
1 * * *
0 X O *
0 1 2
Enter a column between 0 and 2 to play in: 0
2 * * *
1 X * *
0 X O *
  0 1 2
Enter a column between 0 and 2 to play in: 0
2 O * *
1 X * *
0 X O *
  0 1 2
Enter a column between 0 and 2 to play in: 0
Enter a column between 0 and 2 to play in: 0
Enter a column between 0 and 2 to play in: -2
Enter a column between 0 and 2 to play in: 4
Enter a column between 0 and 2 to play in: 1
2 O * *
1 X X *
0 X O *
```

```
      0 1 2
    Enter a column between 0 and 2 to play in: 2
    2 O * *
    1 X X *
    0 X O O
      0 1 2
    Enter a column between 0 and 2 to play in: 2
    2 O * *
    1 X X X
    0 X O O
      0 1 2
    Player 1 Won!
4../connectn.out 1 2 3
    0 * *
      0 1
    Enter a column between 0 and 1 to play in: 0
    0 X *
      0 1
    Enter a column between 0 and 1 to play in: 1
    0 X O
      0 1
    Tie game!
```