ConnectN

Matthew's Stats

Time Taken: 2 hours
Files: 16
Lines of Code: 772

What to Submit

- A zip file containing
    - Your .cpp and .h files that make up your solution
    - A CMakeLists.txt file that will generate an executable named **ConnectN** from your .cpp and .h files
        - In your CMakeLists.txt in the add_executable line, make sure you have `add_executable(ConnectN your_.cpp_files your_.h_files)` so the correctly named executable gets built
- Make sure to zip the files you want to submit and **NOT** the folder that contains the files. Submitting the folder with the files will cause your program to fail to build.

Restrictions and Requirements

- You **MUST** enclose all of your solution, except for main, in a namespace name **ConnectNGame**
- No global variables may be used
- Your submission must contain at least 2 or more .cpp files and one or more .h files
- Your submission must have at least 3 user defined functions in it in addition to main
- You must have at least 2 user defined classes: Board and Player though having more is likely to be very helpful

Description

You are to implement the game of ConnectN. ConnectN is like Connect4 but the user gets to specify the dimensions of the board as well as how many pieces in a row are needed to win. In the game, players take turns dropping a piece into one of the available columns, trying to match N or more of their pieces in a row. If you've never played Connect4 before I recommend that you do so in preparation for this problem. You can play the game here or search online for another version that you like more.

Input

Command Line

- All Command line input will be valid

1. An integer specifying the number of rows on the board
2. An integer specifying the number of columns on the board
3. An integer specifying the number of pieces in a row to win
    a. It is legal for this value to be an amount that would be impossible to get on a board of this size. For example, it is legal to need 10 pieces in a row on a 3 X 4 board.

Standard Input

- Will not always be valid. If invalid input is entered then the user should be continually prompted for input until they enter valid input

User Name

- The user name must be exactly one word long and not the same as the other player's name

Player Piece

- A single character representing the marker for this user that is not the same as the other player's piece
    - When checking to see if the piece is the same as the other player's piece case doesn't matter. For example, if player 1 chose X for their piece player 2 would not be allowed to choose x

Column to Play In

- An integer specifying one of the columns on the board to play in. The column to play in cannot be full

Game Play Loop

Player Turn

- The player selects a column to drop their piece into and drops their piece into it

Ending the Game

- The game ends if either player wins or there is a tie
- A player wins by matching N or more of their pieces in a row either vertically, horizontally, or diagonally anywhere on the board
- A tie occurs if the entire board is filled up without either player winning

Hints

- Start early. You will need every minute of the time you have been given to finish by the deadline
- Play with the sample executable that I've given you on Kodethon to see how the game works and to get an idea of what you need to do before starting
- The Tic Tac Toe game we did in class is a great starting point
- Break things down into classes and functions. Have each class dedicated to a certain idea and have each function do exactly one thing and one thing only.
- Use the top-down programming approach that has been shown in class

Example

In the example below, user input has been underlined to help you distinguish between input and output. You do not have to underline anything.

```
./ConnectN 6 7 4
Player 1, please enter your name: MegaMan
MegaMan, please enter the character you want to use for your piece: M
Player 2, please enter your name: Protoman
Protoman, please enter the character you want to use for your piece:
P
  0 1 2 3 4 5 6
5 * * * * * * *
```

```
4 * * * * * * *
3 * * * * * * *
2 * * * * * * *
1 * * * * * * *
0 * * * * * * *
MegaMan, please enter a column to play in: 1
  0 1 2 3 4 5 6
5 * * * * * * *
4 * * * * * * *
3 * * * * * * *
2 * * * * * * *
1 * * * * * * *
0 * M * * * * *
Protoman, please enter a column to play in: 2
  0 1 2 3 4 5 6
5 * * * * * * *
4 * * * * * * *
3 * * * * * * *
2 * * * * * * *
1 * * * * * * *
0 * M P * * * *
MegaMan, please enter a column to play in: 2
  0 1 2 3 4 5 6
5 * * * * * * *
4 * * * * * * *
3 * * * * * * *
2 * * * * * * *
1 * * M * * * *
0 * M P * * * *
Protoman, please enter a column to play in: 3
  0 1 2 3 4 5 6
5 * * * * * * *
4 * * * * * * *
3 * * * * * * *
2 * * * * * * *
1 * * M * * * *
0 * M P P * * *
MegaMan, please enter a column to play in: 4
  0 1 2 3 4 5 6
5 * * * * * * *
4 * * * * * * *
3 * * * * * * *
2 * * * * * * *
```

```
1 * * M * * * *
0 * M P P M * *
Protoman, please enter a column to play in: 3
  0 1 2 3 4 5 6
5 * * * * * * *
4 * * * * * * *
3 * * * * * * *
2 * * * * * * *
1 * * M P * * *
0 * M P P M * *
MegaMan, please enter a column to play in: 3
  0 1 2 3 4 5 6
5 * * * * * * *
4 * * * * * * *
3 * * * * * * *
2 * * * M * * *
1 * * M P * * *
0 * M P P M * *
Protoman, please enter a column to play in: 4
  0 1 2 3 4 5 6
5 * * * * * * *
4 * * * * * * *
3 * * * * * * *
2 * * * M * * *
1 * * M P P * *
0 * M P P M * *
MegaMan, please enter a column to play in: 5
  0 1 2 3 4 5 6
5 * * * * * * *
4 * * * * * * *
3 * * * * * * *
2 * * * M * * *
1 * * M P P * *
0 * M P P M M *
Protoman, please enter a column to play in: 4
  0 1 2 3 4 5 6
5 * * * * * * *
4 * * * * * * *
3 * * * * * * *
2 * * * M P * *
1 * * M P P * *
0 * M P P M M *
MegaMan, please enter a column to play in: 4
```

```
  0 1 2 3 4 5 6
5 * * * * * * *
4 * * * * * * *
3 * * * * M * *
2 * * * M P * *
1 * * M P P * *
0 * M P P M M *
MegaMan won the game!
```