Program #6

Due Sunday by 11:59pm **Points** 100 **Available** after Nov 19 at 12:01am

Implementing CONNECT 4

Introduction

For this homework project you will have two weeks to complete the programming portion of the assignment. **However, you will be required to submit a design document to Canvas (due on Sunday, November 25th, 11:59pm).** You can use the same documentation template and format that was introduced during week 2. Please review the **recitation syllabus (https://web.engr.oregonstate.edu/~goinsj/resources/CS161/CS161_Recitation_Syllabus.pdf)** (available from the Recitations tab) for additional details on the design document expectations.

Example design document (introduced in week 2): **Polya_template.pdf (https://web.engr.oregonstate.edu/~goinsj/resources/CS161/Polya_template.pdf)**

This particular program implements a game that is played in real life. For this assignment you should try to make your program look as nice as possible.

For program 6, the TAs will compile and grade your code during finals week (and you do not need to schedule a demo time). You must include a ***readme.txt*** *text file in your final submission that provides instructions on how to compile your code. For example, if your code needs to use C++11, then the readme.txt file should tell the TA to compile using the command:*

```
g++ -std=c++11 yourcode.cpp
```

Any other special instructions must also be included in the ***readme.txt*** *file. If you completed any extra credit work, your readme file should also describe that work.*

(85 points) Problem Statement

Write a C++ program that plays the game of CONNECT 4. The game is very simple and you can **view the instructions online here (https://www.hasbro.com/common/documents/dad2614d1c4311ddbd0b0800200c9a66/DE3C8F8050569047F5AA9FBB9F16909B.pdf)**
.
You can also find online implementations if you want to see the game in action. One specific online implementation is at: **https://www.mathsisfun.com/games/connect4.html (https://www.mathsisfun.com/games/connect4.html)**

Your game will allow 1-2 players, and at the end, you need to ask if the user(s) want to play again. Traditionally there are yellow tokens and red tokens. They are inserted into a grid and the first player to get 4 adjacent tokens (horizontally, vertically, or diagonally) wins.

Command Line Arguments

Command line arguments will be used to indicate the number of players as well as the size of the playing grid. The three command line instructions will be provided in the following order: number of players, number of columns, number of rows. A 2 player game implies that two humans will be present and the program will allow each player to take turns. A one player game implies that the human is playing against the computer (see One Player Operation for more details).

Example command to start a 2-player game with 7 columns and 6 rows:

```
./connect_four 2 7 6
```

If you want to start a one player game on a 9 x 7 grid, you would use the following command:

```
./connect_four 1 9 7
```

Two Player Operation

In two player mode, your code will first display the empty grid (with each column numbered across the top). It will then prompt the first player to select a column. After a column is selected, the screen will display the updated game grid with the player's token at the bottom of the selected column. Player two can then choose a column in which to drop their token. This behavior continues (alternating between players) until a winner is determined or until no more tokens can be dropped into the grid (resulting in a tie).

One Player Operation

In one player mode, the human will play against the computer. Your code will first ask the human if they want to have the first move. If so, the human gets to drop the first token. If not, the computer gets to drop the first token. In order to make this programming assignment easier, the computer player does not have to be intelligent. For each computer move, the program will randomly select a column and drop the token. Naturally, the computer player can only drop a token into columns that have at least one empty space remaining. As with the two player operation, game play will alternate between each player.

Additional Requirements

In addition to the earlier specifications, your program must meet these requirements:

- Your program must display the updated game grid after each move.
- If a winner exists, the program must immediately declare the winner and ask if the player(s) want to play again.
- If no more moves are possible and no winner exists (i.e. the entire grid is full of tokens), the program must declare the game a tie and then prompt the player(s) to start a new game.
- Print an error message and recover when the player supplies an invalid column. This could be a column that doesn't exist ("Cat", -4, 142, etc) or it could be a column that is already full of tokens.
- For this program you must validate the incoming command line arguments and ensure that each of the three values is a non-negative number. If an invalid value is provided (negative number, floating point number, text string, etc) then the program must halt execution and display a message to indicate the problem.
- Play the game correctly based on rules and number of players.
- Continue to play until the user selects no.
- You must not have any global variables
- You must use a dynamic 2-dimensional array to represent the grid of tokens.
- The computer player must follow the rules of the game and can only drop tokens in columns that have at least one open space.
- Your functions need to focus on performing a particular task. In other words, you need to use good modular design. If your function uses more than about 20 lines of code, this may be an indication that the code should be split into multiple functions. If the TA notices that your code is not sufficiently modular, you will lose points.
- You must not have memory leaks.
- Segmentation faults are not allowed (e.g. your program crashes).

(10 pts) Extra Credit - Implementing a smarter computer opponent

Instead of using a simple random number generator to select the column, write your computer player implementation so that it has more intelligence. If you choose to attempt this extra credit be sure to document your algorithm in the readme.txt file.

As you can imagine, there are many ways to implement a smarter computer opponent. The options are endless!

(15 pts) Program Style/Comments

In your implementation, make sure that you include a program header in your program, in addition to proper indentation/spacing and other comments! Below is an example header to include. Make sure you review the **style guidelines for this class** **(https://web.engr.oregonstate.edu/~goinsj/resources/general/cpp_style_guideline.pdf)**, and try to follow them, i.e. don't align everything on the left or put everything on one line!

```
/******************************************************
** Program: connect_four.cpp
** Author: Your Name
** Date: 11/22/2018
** Description:
** Input:
** Output:
******************************************************/
```

Don't forget that each of your functions needs to have a header. For example:

```
/******************************************************
** Function:
** Description:
** Parameters:
** Pre-Conditions:
** Post-Conditions:
******************************************************/
```

Assignment Submission

Note that there are two deadlines for this assignment.

Your design document must be scanned and electronically submitted to Canvas (in the assignment tab) by **Sunday, November 25th, 11:59pm**. You may use one of the document scanners (available in KEC1130 and some other College of Engineering labs) to scan your paper design into a PDF document that can be submitted to Canvas.

Electronically submit your C++ source code and the readme.txt file to **TEACH** **(https://engineering.oregonstate.edu/teach)** by the assignment due date on **Sunday, December 2nd, 11:59pm**. Note that your TEACH submission will contain two files.

Special Notes

This is the final programming assignment for this class and you are not required to demo this code. Instead, the TAs will grade your code on their own during finals week. This is why it is important for you to create a readme.txt with any additional information that you feel the TAs should know.

Each program needs to be written according to the **style guidelines** **(https://web.engr.oregonstate.edu/~goinsj/resources/general/cpp_style_guideline.pdf)** for this class. Remember that an important part of computer programming is making sure that your work is easily understandable by other programmers.

Program #6 Rubric

Criteria	Ratings	Pts
Program Header At a minimum, header should contain author's name and a description of the program. (2 pts each) 1 point for submitting code that compiles on flip.		5.0 pts
Good indentation / Use of whitespace Is code easy for the TA to read? Conditional blocks of code should always be indented.		5.0 pts
Each function is documented Every function contains it's own initial block comment (or multiple lines of comments prior to the function definition) that provides the reader with an explanation of the function's purpose. -1 pt for each function that is missing a header		5.0 pts
All functions designed in a modular fashion / No global variables -5 pts if there are any global variables used in the code. If functions are exceptionally long (greater than about 20 lines of actual code) this is a potential indication of poor modularity. -3 pts for each function that the TA concludes is poorly modularized.		12.0 pts
Command Line Functionality (4 pts) Program can run with 1 player or 2 players via command line argument. (3 pts) "connect4 1 4 cat" returns an error and exits properly. (3 pts) "connect4 -1 0 6" returns an error and exits properly.		10.0 pts
Game Operation (5 pts) Program displays the updated grid after each move. (5 pts) Program displays an error message and asks for a valid column if the user enters an invalid choice (e.g. column 14 if there are only 10 columns) (6 pts) When a valid column is selected, a new token (corresponding to the correct player) always appears in the lowest available slot. Subtract 2 of the 10 points for any cases when this does not happen.		16.0 pts
Winner Detection (5 pts) If a winner exists horizontally, the program must immediately declare the winner. (5 pts) If a winner exists vertically, the program must immediately declare the winner. (5 pts) If a winner exists diagonally, the program must immediately declare the winner. (4 pts) If no more moves are possible and no winner exists (i.e. the entire grid is full of tokens), the program must declare the game a tie		19.0 pts
Dynamic Memory Usage Code uses a dynamically allocated 2-dimensional array to represent the grid of tokens. the size of the grid needs to match whatever was provided on the command line. -2 pts if the program does not generate and operate with the correct sized grid.		4.0 pts
Program Functionality Program does not crash during testing (e.g. segmentation fault or similar abrupt halts) (6 pts). User can choose to replay the game after winning or losing (2 pts).		8.0 pts
Computer Opponent If the user plays in 2-player mode, the computer always drops a token in one of the columns that is not full. Specifically test the case when one column or more columns are full. -3 pts for each incidence of faulty behavior		6.0 pts
Memory Management (see details) Valgrind does not report any errors about the code reading or writing from memory that is not allocated. Also, there are no warnings saying that a conditional jump depends on an uninitialized value. In other words, you must properly allocate your dynamic memory before utilizing it.		5.0 pts
Memory Cleanup Running Valgrind on the program reports that "All heap blocks were freed" (or something indicating that all memory is freed). There should be no blocks reported as "in use" at exit or "possibly lost". Note that this is not the same as the memory management category.		5.0 pts

Criteria	Ratings	Pts
Extra Credit - Implementing a smarter computer opponent (10 pts) The user implements a more intelligent computer opponent (something more advanced than the random column selection algorithm). The algorithm must be documented in the readme.txt file in order to receive credit!		0.0 pts
Total Points: 100.0		