Lab 8, Chapter 8, Connect-X game.

Search online for a game called connect 4; see links: https://www.mathsisfun.com/games/connect4.html or https://kevinshannon.dev/connect4/. These games create a two-dimensional game board. The size of the game board can vary, but is often about a small amount taller and/or wider than the number of tokens needed to form a winning line. Using these links, play a game of connect 4 to get an idea of how the game works. The players alternate moves by indicating the column to drop a token into. The token "falls down" from the top into the board, dropping as far as it can, to the bottom or falling onto another piece already on the board.

Lab 8: Write an object-oriented, modular program that allows two players to play a game on a two -dimensional board. The object of the game is to get several tokens together in a line. The line may be horizontal, vertical, or diagonal. The game board should be a class. Use a two-dimensional array (or vector) to store the state of the game board as private data within the "Board" class. Use methods on the class: setters and getters, to make moves on the board inside the class. The constructor of class Board should create an empty game, ready to play.

You decide several things about the game:

1) The dimensions of the board, with a minimum size of 3 x 3
2) The length of the line required to win the game, with a minimum length of 3
3) The tokens, or "markers" on the board, with two symbols used, one for each player
4) How to indicate which column to drop the marker into.

You must use a programmer-defined class data-type for this lab, with constructor(s), setter(s) and getter(s)! Alter the state of the board using public member functions of class Board. Submit one file: DLS_L8_Lastname.cpp. Replace DLS with the course designator, and Lastname with your last name.

The following example uses a minimal size of a 3 x 3 board, 3 in a line to win, uusing column numbers 1, 2, 3 at the top to indicate the move. This example shows the default player symbols 'X' and 'O'. **In your lab, do NOT use the letters 'X' and 'O'.** Instead, use the first letter from your first and first letter from your last name. If these letters are the same, choose a different letter from your last name. For example, Abigail Brown would be playing letter 'A' against letter 'B'; Wilbur Wilson plays letter 'W' against letter 'I'. Assuming letters 'X' and 'O', game play looks like this:

```
Welcome to connect 3. Below is the initial game board.
Select column number 1, 2, 3; along the top; to indicate your move.
The two player tokens are X and O. X begins.

    123
    ---
    ***
    ***
    ***

Player 1(X) enter your column to drop into (1-3, zero to quit): 1
    123
    ---
    ***
    ***
    X**

Player 2(O) enter your column to drop into (1-3, zero to quit): 1
    123
    ---
    ***
    O**
    X**
```

```
Player 1(X) enter your column to drop into (1-3, zero to quit): 2
    123
    ---
    ***
    O**
    XX*

Player 2(O) enter your column to drop into (1-3, zero to quit): 3
    123
    ---
    ***
    O**
    XXO

Player 1(X) enter your column to drop into (1-3, zero to quit): 2
    123
    ---
    ***
    OX*
    XXO

Player 2(O) enter your column to drop into (1-3, zero to quit): 3
    123
    ---
    ***
    OXO
    XXO

Player 1(X) enter your column to drop into (1-3, zero to quit): 2
    123
    ---
    *X*
    OXO
    XXO

Congratulations! Player 1(X) has won the game!!!

Do you want to play again? (y/n): n

Player 1 won: 1 times. 100.00%  Player 2 won: 0 times. 0.00%
This ends the Connect game. Goodbye.
```

Basic: Start out by displaying an empty game board, and ask for the move of player one. Redraw the board after every move. Alternate moves between player one and player two. Stop the game when no legal moves are available. Do not allow invalid moves: invalid column, or a move onto a full column. No checking for win/lose or keeping score. Allow the player to stop the game immediately. **(max 25 points).**

Medium: After each move, determine the game status. Either: player 1 won, player 2 won or nobody won. If nobody won, determine whether play can continue (more places open) or play is over because the board is full and nobody won (a tie, called a "cat's game").

To determine if a game is won, check if there are enough tokens in a line. Check rows, columns, and diagonals. For example, on a 3 x 3 board, the minimal size, there are 8 ways to win with 3 in a line: top row, middle row, bottom row; left column, middle column, right column, or top-left to bottom-right diagonal or top-right to bottom-left diagonal. If nobody won, and there are no open spots, you have a tie, a cat's game. There are many ways to determine the games status: win / loss / tie (cat’s game). **(max 28 points).**

Full: Play as many games as the user desires. Keep score of how many games each player won. If a player quits the game early, the game is forfeit, and that player loses that game. After the game, ask the user if he/she wants to continue. Provide percentage of win/loss/cats for both players**. (max 30 points).**

Extra credit option 1: Provide a different color for each player's token. You can change the color of characters displayed - see page 374 (7ed), page 392 (8ed), page 392 (9ed), page 396 (10ed). This option is available only for Windows environments. Use an enumerated type to name the colors and avoid "magic numbers". This makes the code more self-documenting. **(max 32 points: 2 extra for using color.).**

Extra credit option 2: Enhance the game so the player can play another player, as described above, OR play the computer. The computer can use a random number generator to generate legal moves. Can you make the computer "smarter" than just generating random valid moves? Online games offer various levels of how "smart" the computer may be when playing the game. This option is available for both Windows and non-Windows environments. **(max 32 points: 2 extra for player 2 being the computer.).**

Further consideration:
Consider having the computer play itself, without any input or output during the game. The program keeps track of game results. What if the computer plays itself 10 times? 100 times? 1000 times? 1 billion times? How many times did computer player 1 win? Computer player 2? How many cat's games? What does this tell you about the game? This method is called Monte Carlo simulation, and allows programmers to develop winning game strategies.