Programming Project

Instructions:

- **The project requires completing a set of tasks by April 20, 2023.**
- **Your objective is to create a connect4 game, which is played with two players and uses a 6 by 7 (row by column) grid. Each player takes a turn inserting one of their pieces into the grid by releasing it from the top of a selected column. The first person to get four of their pieces in a row vertically, horizontally, or diagonally wins.**
- **The display of the grid must be as follows**

```
  0  1  2  3  4  5  6
5 [ ] [ ] [ ] [ ] [ ] [ ] [ ]
4 [ ] [ ] [ ] [ ] [ ] [ ] [ ]
3 [ ] [ ] [ ] [ ] [ ] [ ] [ ]
2 [ ] [ ] [ ] [ ] [ ] [ ] [ ]
1 [ ] [ ] [ ] [ ] [ ] [ ] [ ]
0 [ ] [ ] [ ] [ ] [ ] [ ] [ ]
```

  where the above illustration is of an empty grid.

- **You must use tokens `'O'` and `'X'` to represent the pieces of each player.**
- **To receive any credit, the accompanying CPP file, `"main.cpp"`, must compile without any errors and modifications.**
- **The class must be written in a header file named `"Connect4.h"` and the class must be named *Connect4*.**
- **The header file can only use the libraries *iostream*, *string*, *sstream*, *cctype*, *iomanip*, and *cmath*.**
- **All fields of *Connect4* must be private, but there is no limitation on how many fields can be included in the class. However, a comment must be provided for each field that explains its purpose.**
- **Only the methods listed below should be public, but there is no limitation on how many additional methods can be included in the class. However, a comment must be provided for each additional method that explains its purpose.**
- **Methods used for validating or determining an aspect of the game must be constant methods.**
- **No additional files can be created for the project.**
- **Points for a task will only be awarded if the task is completed accurately.**
- **Cheating of any kind is prohibited and will not be tolerated.**
- **Violating and/or failing to follow any of the rules will result in an automatic zero (0) for the project.**

Grading

Problem	Maximum Points	Points Earned
01	0.5	
02	0.5	
03	0.5	
04	0.5	
05	1	
06	3	
07	1	
08	1	
09	1	
10	1	
Total	10	

1. Define the default constructor that initializes the game.
2. Define the copy constructor.
3. Define the assignment operator.
4. Define the destructor.
5. Define a bool constant method named `HasWinner()` that takes no parameters and returns true only if a winner has been decided.
6. Define a bool method named `MakeMove()` that takes an int parameter. If the parameter represents a valid column [a value between 0 and 6 inclusively], it determines if the column has space. If the column has space, it adds the appropriate piece to the board, and then, determines if the current player is a winner and returns true; otherwise, it returns false. Additionally, if the current player is not a winner, it changes the current player. This method **must not** modify any fields or perform any comparisons of fields directly.
7. Define a void method named `Reset()` that takes no parameters and resets the game to its initial state.
8. Define a char constant method named `CurrentPlayer()` that takes no parameters and returns the token of the current player.
9. Define a string constant method named `ToString()` that takes no parameters and returns a string of the board in the same format as in the instructions.
10. Define a friend ostream operator that returns its output in the same format as `ToString()`.

Hints & Suggestions

- Represent the board as a string array with a size of 6 such that each element of the array has a length of 7.
- Use a special character to represent blanks on the board such as an asterisk. Convert it to a space only when displaying the board.
- Declare an array that keeps track of the number of pieces that are in each column. Doing so will make it easy to determine the position of the new piece added to the grid.
- Declare a static constant array field that holds the token characters of the players.
- When determining a winner, separately analyze if there is a winner horizontally, vertically, and diagonally. Furthermore, only check if there is a winner in the horizontal, vertical, and diagonal lines that the current piece is in.