The American University in Cairo
Computer Science and Engineering Department
CSCE110101/02

Assignment 2

Submission Instructions:

- Submit a **zip file** containing a report and all the cpp and .h files. (it's ok to submit 1 cpp file and a report)
- Submit a **pdf report** file with screenshots of your outputs, any errors you still have, or any remarks you want to add.

By submitting this assignment, I affirm that I have followed AUC's Code of Academic Ethics and the work submitted is my own. I have not consulted unauthorized resources or materials nor collaborated with other individuals unless allowed.

GameToGo is an online gaming startup that offers a collection of engaging and interactive board games for players of all ages. Their platform focuses on providing a virtual environment for playing classic board games like Tic-Tac-Toe and Connect Four. They emphasize user experience, fair play, and community engagement.

In this assignment, you will play the role of a developer at GameBoardGo. You will design and implement two classic board games, Tic-Tac-Toe and Connect Four, within their online gaming platform. These games should be implemented using object-oriented principles such as inheritance and polymorphism. Additionally, you will utilize operator overloading in your implementations. Your mission is to create a seamless and enjoyable gaming experience for their users.

Part 1: Base Board Class (20 points)

- Create a base class called BoardGame which represents the common attributes and behaviors shared among all board games on GameToGo.
- Define a constructor for the BoardGame class that initializes the dimensions of the game board (rows and columns).
- Implement the method called printBoard in the BoardGame class. This method should print the current state of the game board to the user's interface.
- Create a method named makeMove in the BoardGame class. This method should allow a player to make a move on the game board. Ensure that you handle move validation and maintain turn order.
- Overload the operator + to do the same functionality as makeMove in the BoardGame class.
- Implement a pure virtual method named isGameOver in the BoardGame class. This method should check whether the game is over due to a win, draw, or other game-specific conditions. Override this method in derived game classes to provide game-specific win and draw conditions.

Part 2: Tic-Tac-Toe Game (20 points)

- Create a class named **TicTacToe** that inherits from the BoardGame class. This class will represent our virtual Tic-Tac-Toe game.
- Implement the print method in the TicTacToe class, which should render the Tic-Tac-Toe board with 'X' and 'O' marks.

- Override the makeMove method in the TicTacToe class to allow players to make moves, alternating between 'X' and 'O'. Ensure that you handle move validation and win conditions.
- Implement the game logic to check for a win or a draw. Override the isGameOver method in the TicTacToe class to determine the game's outcome.

Part 3: Connect Four Game (20 points)

- Create a class named **ConnectFour** that inherits from the BoardGame class. This class will represent our virtual Connect Four game.
- Implement the print method in the ConnectFour class, which should render the Connect Four board with 'X' and 'O' marks.
- Override the makeMove method in the ConnectFour class to allow players to make moves, following Connect Four's rules. Ensure that you handle move validation and win conditions.
- Implement the game logic to check for a win or a draw. Override the isGameOver method in the ConnectFour class to determine the game's outcome.

Part 4: Player Class (20 points)

- Create a class named **Player** to represent a player in a board game. The Player class should have the following attributes:
    - A name to store the player's name (as a string).
    - A symbol to store the player's symbol (e.g., 'X' or 'O' or ‘Red’ or ‘Blue’).
    - Overload the << operator to allow printing a player's information using std::cout. When you use std::cout << player, it should print the player's name and symbol.
    - Integrate the Player class into the board class. Modify it to have instances of two Player objects, one for each player. Update the game logic to use these players and print the name of the player whose turn it is to make a move

Part 5: Main Program (20 points)

Create a main program that demonstrates the functionality of the two games.

Bonus (15 points): Make the games look good using QT.