```
/**
-------------------------------------------------------------------------------------------
----------------------
* This program allows two people to play the game connect 4. The object of this game
is to place 4 of your pieces in a row in the game board. The 4 pieces can be either
horizontal, vertical, or diagonal. The player gets to choose a column and their token
slides down the column until it hits the bottom or it lands on another piece.
* Class: CS 141, Fall 2023
* System: ZyBook Lab *
* @author Safa Patel
* @version November 17, 2023
-------------------------------------------------------------------------------------------
-------------------------*/
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;

// Function to display the current game board
// Receives a 2D array representing the game board
// Does not return anything
void displayBoard(char grid[6][7])
{
   // Code for displaying the game board
   cout << "0 1 2 3 4 5 6 " << "\n";
   for (int row = 0; row < 6; row++)
   {
       cout << "\t";
       for (int column = 0; column < 7; column++)
       {
           cout << grid[row][column] << " ";
       }
       cout << endl;
   }
}

// Function to check if a move is valid in a given column
// Receives a column index and a 2D array representing the game board
// Returns a boolean indicating whether the move is valid or not
bool validMove(int column, char grid[6][7])
{
   // Code for checking if the move is valid
```

```
    return column >= 0 && column < 7 && grid[0][column] == '-';
}

// Function to make a move in a given column
// Receives a column index, a player token, and a 2D array representing the game board
// Does not return anything
void makeMove(int column, char player, char grid[6][7])
{
    // Code for making a move in the specified column
    for (int row = 6 - 1; row >= 0; row--)
    {
        if (grid[row][column] == '-')
        {
            grid[row][column] = player;
            return;
        }
    }
}

// Function to check if a player has won the game
// Receives a player token and a 2D array representing the game board
// Returns a boolean indicating whether the player has won or not
bool checkWin(char player, char grid[6][7])
{
    // Check horizontal
    for (int row = 0; row < 6; row++)
    {
        for (int col = 0; col < 7 - 3; col++)
        {
            if (grid[row][col] == player && grid[row][col + 1] == player &&
                grid[row][col + 2] == player && grid[row][col + 3] == player)
            {
                return true;
            }
        }
    }

    // Check vertical
    for (int row = 0; row < 6 - 3; row++)
    {
        for (int col = 0; col < 7; col++)
        {
```

```
            if (grid[row][col] == player && grid[row + 1][col] == player &&
                grid[row + 2][col] == player && grid[row + 3][col] == player)
            {
                return true;
            }
        }
    }

    // Check diagonal (down-right)
    for (int row = 0; row < 6 - 3; row++)
    {
        for (int col = 0; col < 7 - 3; col++)
        {
            if (grid[row][col] == player && grid[row + 1][col + 1] == player &&
                grid[row + 2][col + 2] == player && grid[row + 3][col + 3] == player)
            {
                return true;
            }
        }
    }

    // Check diagonal (down-left)
    for (int row = 0; row < 6 - 3; row++)
    {
        for (int col = 3; col < 7; col++)
        {
            if (grid[row][col] == player && grid[row + 1][col - 1] == player &&
                grid[row + 2][col - 2] == player && grid[row + 3][col - 3] == player)
            {
                return true;
            }
        }
    }
    return false;
}

// Function to check if the game board is full
// Receives a 2D array representing the game board
// Returns a boolean indicating whether the board is full or not
bool boardFull(char grid[6][7])
{
    // Code for checking if the board is full
```

```
    for (int col = 0; col < 7; col++)
    {
        if (grid[0][col] == '-')
        {
            return false;
        }
    }
    return true;
}

// Function to check if a column is full
// Receives a column index and a 2D array representing the game board
// Returns a boolean indicating whether the column is full or no
bool columnFull(int column, char grid[6][7])
{
    // Code for checking if the column is ful
    if (grid[0][column] == '-')
    {
        return false;
    }
    return true;
}

// Main function for the Connect 4 game
// Manages the overall game logic, including setup, user input, and game state updates
// Does not receive any parameters
// Returns an integer indicating the program's exit status
int main()
{
    // Introduction and game setup
    cout << "This is the Game Connect 4." << endl;
    cout << "Each player should place an X or an O in the space " << endl;
    cout << "by entering the column you want to place the piece." << endl;
    cout << "The piece will fall until it reaches the bottom or " << endl;
    cout << "the current pieces in the board. When X or O gets 4 in " << endl;
    cout << "a row (either horizontally, vertically, or diagonally, "  << endl;
    cout << "then that person wins. The user can enter Q (or q) to " << endl;
    cout << "end the game early." << endl;
    cout << "Let's get started!!!" << endl;

    // Initialize the game board with empty spaces
    char grid[6][7];
```

```
for (int row = 0; row < 6; row++)
{
    for (int col = 0; col < 7; col++)
    {
        grid[row][col] = '-';
    }
}

// Set current player to 'X' and initialize game over to false
char currentPlayer = 'X';
bool gameover = false;

// Main game loop
while (!gameover)
{
    // Display empty game board before any moves
    displayBoard(grid);

    // Take in user move for which column they want to put their piece
    int column;
    cout << "It is " << currentPlayer << "'s turn." << endl;
    cout << "Enter a column to place your piece: ";
    string input;
    cin >> input;
    cout << endl;

    // Check if the player wants to quit the game
    if (input == "Q" || input == "q")
    {
        cout << "Ending Game" << endl;
        break;
    }

    // Check if the chosen column is already full
    if (columnFull(column, grid))
    {
        cout << "column chosen is already full" << endl;
    }

    // Convert user input of a string number to an int
    column = stoi(input);
```

```
        // Manages the game's turn-based logic, makes sure the moves are valid, and
checks for a win or a draw, ultimately updating the game state
        if (column >= 0 && column <= 6 && validMove(column, grid))
        {
            makeMove(column, currentPlayer, grid);

            if (checkWin(currentPlayer, grid))
            {
                displayBoard(grid);
                cout << endl;
                cout << "Game is Over, Player " << currentPlayer << " got 4 in a
row!!!! " << endl;
                gameover = true;
            }
            else if (boardFull(grid))
            {
                displayBoard(grid);
                cout << "Board is Full, It's a Draw!!!" << endl;
                gameover = true;
                break;
            }
            else
            {
                currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';
            }
        }
        else
        {
            cout << "Please enter a valid column" << endl;
        }
    }

    return 0;
}
```