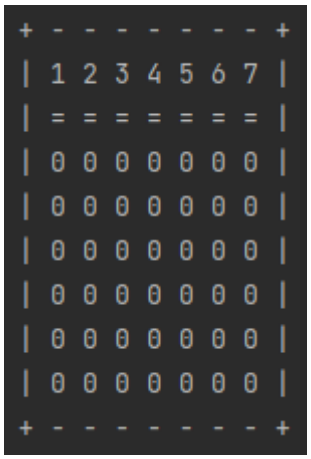
EE1010 **Connect 4 (Report)** **Priyam Patel**

User manual

This Connect 4 is a text-based recreation of the physical game Connect 4. In Connect 4 two players take turns selecting a column to place their piece. In this case, the piece is either a R for Player 1's piece or a Y for Player 2's piece. The piece then drops down into the lowest row on the game board. A win is determined when four 1s or four 2s are in either a vertical, horizontal, positive diagonal, or negative diagonal pattern with no breaks. A tie is determined when every slot on the game board has been filled without four simultaneous pieces in a row.

Instructions:

- Player VS Player
  - Click the "Run" button on the right top of the screen
  - Next users will be on the main menu of the game where they can choose what game mode they would like to play
  - Player 1 will start and be asked to select a column (1 -7)
  - Player 2's turn will happen after Player 1's turn
  - The game will continue going back and forth until a win or a tie is determined
  - The game will stop after a win or tie happens
  - If it's a tie the game will ask if both players would like to play again if not, then game ends.



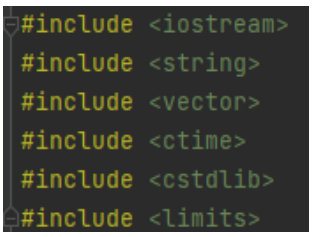
- Player VS Computer
  - In this game mode it is the same as vs a player but instead the user will be playing against the computer.

Code development

The sections below are the main developments of my game, as they required the bulk of the development process to make a full functioning text-based connect 4.

Libraries:

- I got all the libraries needed for my code to work
- **Iostream** provided basic input and output services for C++ programs. iostream used the objects cin and cout, for sending data to and from the standard streams input, output, error (unbuffered), and log (buffered) respectively.
- **String** provided the std::string variety which is a lot easier to use than then old C-style "strings". Using #include <string> enabled my game to get the correct information needed
- **Vectors** were the dynamic arrays that I used to store data. Unlike arrays, which are used to store sequential data and are static in nature, Vectors provide more flexibility to the program when making the game board hence I opted to use vectors instead of arrays to make the game board.



- **Limits** is for a Numeric limits type, and it provides information about the properties of arithmetic types (either integral or floating-point) in the specific platform for which the library compiles.

Main menu:

- In the main menu I used the char function to determine what game mode the user entered and give an appropriate output.

```
void welcome (char& level) { // Welcome Message and Main Menu...

    cout << "\n-------------------------------------------------------------";
    cout << "\n|| Welcome to the Connect 4 Game || MADE BY PRIYAM PATEL ||";
    cout << "\n-------------------------------------------------------------";
    cout << "\n||\t      Chose whom you'd like to play against:     ||\n";
    cout << "||                                                       ||";
    cout << "\n||\t      Type 'P' for Real Player                   ||";
    cout << "\n||\t      Type 'C' for Computer                      ||\n";
    cout << "||                                                       ||";
    cout << "\n||\t      Or                                         ||\n";
    cout << "||                                                       ||";
    cout << "\n||\t      Type 'I' for Instructions                  ||\n";
    cout << "||                                                       ||";
    cout << "\n-------------------------------------------------------------";
    cout << "\n\t      --> Please enter your choice:";

    cin >> level;
}
```

Game board:

- I used vectors instead of arrays to make the game board as it allowed me to make the game board using less lines of code, which optimises the game.

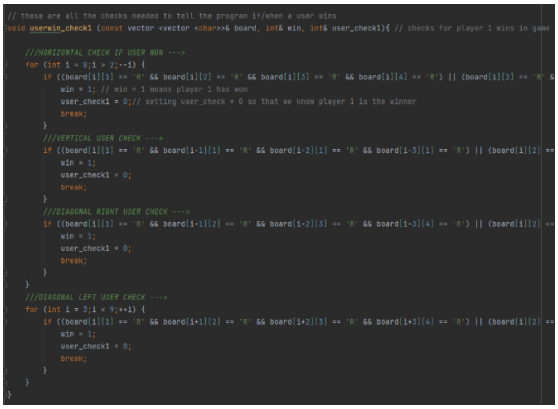
```
void display (vector <vector <char>>& board) { // Displays the game board
    for(int h = 0; h < 10; ++h) { // horizontal print
        for(int v = 0; v < 9; ++v) { //vertical print
            cout << board[h][v] << " ";
        }
        cout << "\n";
    }
}
```

```
// makes the empty game board ready to play
vector<vector<char>> board = {

        {'+', '-', '-', '-', '-', '-', '-', '-', '+'},
        {'|', '1', '2', '3', '4', '5', '6', '7', '|'},
        {'|', '=', '=', '=', '=', '=', '=', '=', '|'},
        {'|', '0', '0', '0', '0', '0', '0', '0', '|'},
        {'|', '0', '0', '0', '0', '0', '0', '0', '|'},
        {'|', '0', '0', '0', '0', '0', '0', '0', '|'},
        {'|', '0', '0', '0', '0', '0', '0', '0', '|'},
        {'|', '0', '0', '0', '0', '0', '0', '0', '|'},
        {'|', '0', '0', '0', '0', '0', '0', '0', '|'},
        {'+', '-', '-', '-', '-', '-', '-', '-', '+'},
};
```

All the checks required to check when someone wins the game:

- This part of the code makes sure that when the game is being played and someone gets a four in a row, it tells the program that there is a winner and goes on to tell the winner announcing part of the code to display the winner



```
// these are all the checks needed to tell the program if/when a user wins
void userwin_check1 (const vector <vector <char>>& board, int& win, int& user_check1){ // checks for player 1 wins in game

    ///HORIZONTAL CHECK IF USER WON --->
    for (int i = 8;i > 2;--i) {
        if ((board[1][i] == 'R' && board[1][i-1] == 'R' && board[1][i-2] == 'R' && board[1][i-3] == 'R') || (board[1][i-1] == 'R' &
            win = 1; // win = 1 means player 1 has won
            user_check1 = 0;// setting user_check = 0 so that we know player 1 is the winner
            break;
        }
        ///VERTICAL USER CHECK --->
        if ((board[1][i] == 'R' && board[i-1][i] == 'R' && board[i-2][i] == 'R' && board[i-3][i] == 'R') || (board[i][i-1] ==
            win = 1;
            user_check1 = 0;
            break;
        }
        ///DIAGONAL RIGHT USER CHECK --->
        if ((board[1][i] == 'R' && board[i-1][i-1] == 'R' && board[i-2][i-2] == 'R' && board[i-3][i-3] == 'R') || (board[i][i-1] ==
            win = 1;
            user_check1 = 0;
            break;
        }
    }
    ///DIAGONAL LEFT USER CHECK --->
    for (int i = 3;i < 9;++i) {
        if ((board[1][i] == 'R' && board[i+1][i-1] == 'R' && board[i+2][i-2] == 'R' && board[i+3][i-3] == 'R') || (board[i][i-1] ==
            win = 1;
            user_check1 = 0;
            break;
        }
    }
}
```

Computer thinking where to put the piece:

- This part of the code is the computer thinking where to put its piece
- Once the player enters where they want to put their piece
- The computer will set random and pick a number between 1 and 7
- And where its possible to put down it will do so
- After letting the player know where is put the piece

```
display(board); // shows the updated board
userwin_check1(board, win, user_check1); // calls in function that checks if player 1 or computer has won the game...
if (win != 1) { //only do it when player 1 has not won the game already
    srand( Seed: time( Time: nullptr)); // sets random seed
    comp_entry = rand() % 7 + 1; // computer selects randomly
    comp_fun_call(comp_entry, row_1, row_2, row_3, row_4, row_5, row_6, row_7,board); // calls the computer function which places into the board
    cout << "The computer has chosen to fill column " << comp_entry << ".\n"; // lets player 1 know where the computer put its piece
    counter++; // adding counter for draw check
    if (counter == 42) { // if counter is 42 and no one has won then board is full
        win = 4; // set win to 4 which will draw the game
        user_check1 = 0; // sets player 1 check to 0
    }
}
compwin_check(board, win, user_check1); // calling function that checks if the computer won...
```

Instructions:

- This part of the code will display simple instructions on how to play the game
- Once the user has read how to play
- It will ask of they are ready to play now or not
- If yes, then it will take them to the main menu where they can use what game mode they would like to play
- If no, then it will end the program

```
else if(level == 'I' || level == 'i'){ // if user uses to learn how play then this will be executed
    cout << "\n\t\t|| How to Play ||\n";
    cout << "\n\t|| Player VS Player mode ||\n";
    cout << "\n\t--> In this mode there will be two players (Player 1 and Player 2)\n";
    cout << "\t--> Starting with the game with player 1 which will be indicated as R for RED on the game board\n";
    cout << "\t--> And Player 2 indicated as Y for YELLOW on the game board\n";
    cout << "\t--> Both players will take turns to play\n";
    cout << "\t--> Whichever player gets a connect 4 either vertically, horizontally, or diagonally first will WIN\n";
    cout << "\t--> If its a DRAW you will get an option to play again.\n";
    cout << "\n\t|| Player VS Computer mode ||\n";
    cout << "\n\t--> In this mode you will be playing against the computer\n";
    cout << "\t--> Starting the game with player 1 which will be indicated as R for RED on the game board\n";
    cout << "\t--> And the Computer indicated as Y for YELLOW on the game board\n";
    cout << "\t--> Once Player 1 has taken their turn the computer will go next\n";
    cout << "\t--> And this will be repeated until either the player or computer\n";
    cout << "\t--> Gets a connect 4 either vertically, horizontally, or diagonally first will WIN\n";
    cout << "\t--> If its a DRAW or if the computer WINS you will get and option to play again.\n\n";
    ready_to_play(flag, user_entry1, user_entry2, comp_entry, row_1, row_2, row_3, row_4, row_5, row_6, row_7, counter); // asks the user if they want to play the game now or not
}
else { // if the options 'P', 'C' and 'I' where not picked then executes the below
    cout << "Invalid level choice. Please try again."; //gives error message
    flag = 1; // flag = 1 so we loop again
}
```

EE1010 **Connect 4 (Report)** **Priyam Patel**

Sources:

- https://www.cplusplus.com/reference/cstdlib/srand/
- https://www.codespeedy.com/taking-only-integer-input-in-cpp/
- https://www.cplusplus.com/reference/vector/vector/
- https://www.w3schools.com/cpp/cpp_data_types_char.asp

Reflection

If I were to start this project again, then,

- These are the features I would keep the same:
  - The game board
  - The main menu
  - Both the game modes
- These are the features that I would like to add if I did make the game again:
  - Some sort of leader board system
  - The vs computer game mode to be AI
  - This would allow me to add different levels of difficulty when playing against the computer

Module evaluation and suggestions

Things I have learnt in this module:

- How to program in C++
- How to write/program a whole game
- And all the basic things linked to learning to code in C++, to name a few:
  - arithmetic, strings, and decisions
  - loops
  - functions and other tools
  - arrays
  - data sorting
  - structures and pointers

Things which should be taught differently next year:

- Have more real-world applications, where we can apply what we learn to real world problems.

Topics I wanted to learn:

- Coding in python
- How to make databases
- Amateur AI
- Amateur machine learning

Overall feedback:

- All in all, this module was great and was well taught/delivered by the lecturers, and a lot was learnt. Given I had no previous experience coding in C++ to making a full text-based game in C++ says a lot itself.