PROJECT MANUAL:

C-o-n-n-e-c-t-4

Introduction:

Connect 4 is a classic two-player strategy game where the objective is to be the first to form a line of four of your own game pieces in a row, either vertically, horizontally,(or diagonally) on a 6x7 grid.

Components:

- **Grid:** The game is played on a grid of 6 rows and 7 columns.
- **Players:** Two players take turns placing their markers on the grid.
    - ❖ Player 1 uses **'X'**
    - ❖ Player 2 uses **'O'**

Game Setup:

1. **Starting the Game:** The game begins with an empty grid.
2. **Names:** Players enter their names at the start of the game.

Gameplay:

1. **Turns:** Players take turns to drop their pieces into a column of their choice.
2. **Objective:** The goal is to create a line of four of your own pieces either vertically, horizontally, (or diagonally.)
3. **Winning:** The first player to achieve this wins the game.
4. **Draw:** If all columns are filled without either player achieving four in a row, the game results in a draw.

Instructions:

Instructions ('I' Key):

- Press 'I' to learn how to play Connect 4.
- Players alternate turns to place pieces on the grid.
- The first player to get four of their pieces in a row wins.
- Rows can be vertical, horizontal,(or diagonal.)

Rules ('R' Key): Press 'R' to review the game rules.

- Player 1 uses 'X' and Player 2 uses 'O'.
- The grid is 6 rows by 7 columns.

Start Game ('P' Key):

- Press 'P' to start playing after reviewing instructions and rules.

Game Interface:

- **Grid Display:** The grid is displayed with column numbers **(0-6)** above it for reference.
- **Markers:** **'X'** and **'O'** are used to represent the players' moves on the grid.
- **Status Updates:** The game informs players when it's their turn, **prompts for column choices**, and announces the winner or if it's a draw.

End of Game:

- **Winning:** The game ends when a player successfully connects four of their markers.
- **Draw:** If the grid is full and neither player has four in a row, the game ends in a draw.

Conclusion:

Connect 4 is a simple yet engaging game that tests players' strategic thinking and spatial reasoning. Enjoy the challenge of your opponent to connect four in a row!

DRY RUN OF CONNECT 4 GAME CODE

Initialization and Function Declarations

Global Variables:

- const int ROW = 6;
- const int COL = 7;
- char arr[ROW][COL];
- int i, j, col;

Function Declarations:

- void mannual();
- void instruction();
- void grid();
- bool insertInColumn(int col, char player);
- bool gameover(char player);
- bool draw();

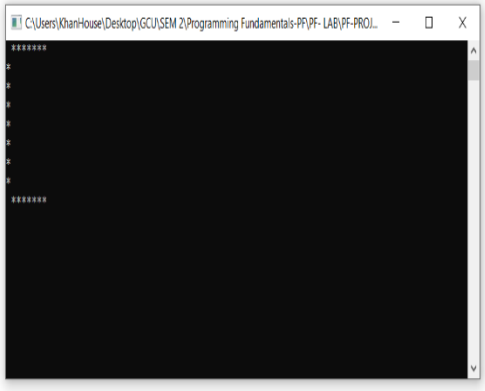
Functions Explanation

FUNCTION#1

mannual() Function Execution:

This function will execute at the start of the game and Outputs a stylized welcome message and instructions for the game. Through escape sequences we have built the whole design of our game “CONNECT

```
include<iostream>
sing namespace std;
nt main()
{
out<<" ******* "<<endl;
   cout<<"* "<<endl;
   cout<<"*"<<endl;
   cout<<"*"<<endl;
   cout<<"*"<<endl;
   cout<<"*"<<endl;
   cout<<"*"<<endl;
   cout<<" ******* "<<endl;
   cout<<endl<<endl;
   cout<<endl<<endl;
   cout<<endl<<endl;
   cout<<endl<<endl;
```



Example of the Code of the Manual Function

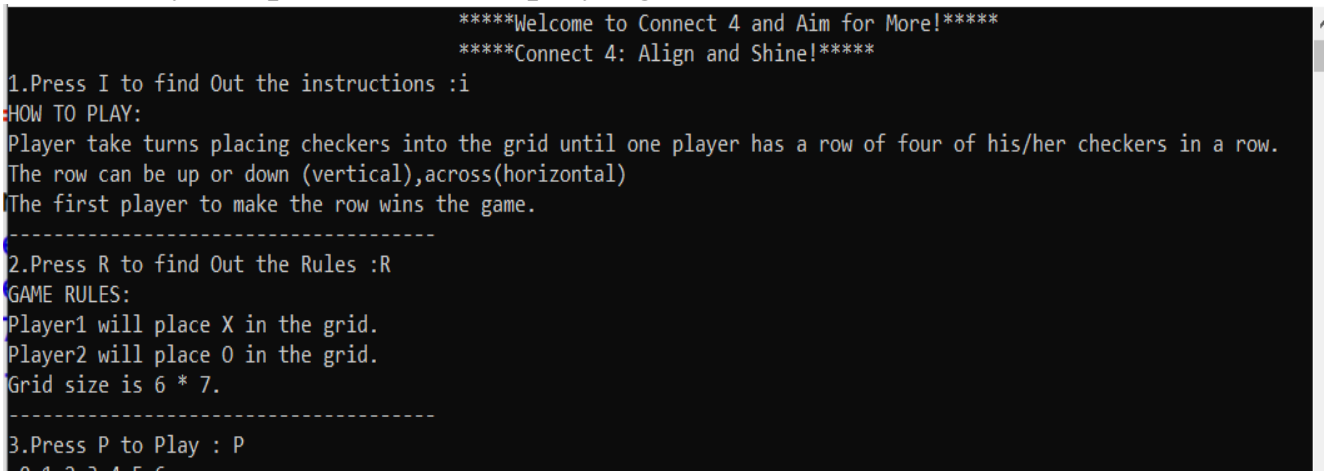
4".

FUNCTION#2:

`instruction()`

Prompts the user for instructions:

- Player inputs 'I' to view gameplay instructions.
- Player inputs 'R' to view game rules.
- Player inputs 'P' to start playing.



- Menu of the game in which if/else statements are used to show instructions and rules of the game is created. As it is a multiplayer game, two players 'X' and 'O' will play simultaneously.
-

FUNCTION#3

`grid()`

Iteration	i	j	Action	output
Initial	-	-	Function grid() is called.	
	-	-	Header and Separator Output:	
	-	-	cout << " 0 1 2 3 4 5 6" << endl;	`0 1 2 3 4 5 6`

	-	-	cout << " _ _ _ _ _ _ _ _" << endl;	`_ _ _ _ _ _ _ _`
Outer LOOP	0	-	for (int i = 0; i < ROW; i++)	
Inner LOOP	0	0	for (int j = 0; j < COL; j++)	
	0	0	cout << "|"	<< (arr[0][0] ? arr[0][0] : ' ');
	0	-	Inner loop completes for j = 0 to COL-1.	
	0	-	cout << "|"<<endl	
	0	-	cout << " _ _ _ _ _ _ _ _" << endl;	_ _ _ _ _ _ _ _"
Outer LOOP	1	-	for (int i = 0; i < ROW; i++)	
Inner LOOP	1	0	for (int j = 0; j < COL; j++)	
	1	0	cout << "|"	<< (arr[1][0] ? arr[1][0] : ' ');
	1	1	cout << "|"	<< (arr[1][1] ? arr[1][1] : ' ');
	1	-	Inner loop completes for j = 0 to COL-1.	
	1	-	cout << "|"<<endl	
	1	-	cout << " _ _ _ _ _ _ _ _" << endl;	_ _ _ _ _ _ _ _"
Outer LOOP	2		for (int i = 0; i < ROW; i++)	
Inner LOOP	2	0	for (int j = 0; j < COL; j++)	

	2	0	cout << "|"	<< (arr[2][0] ? arr[2][0] : ' ');
	2	1	cout << "|"	<< (arr[2][1] ? arr[2][1] : ' ');
	2	2	cout << "|	<< (arr[2][2] ? arr[2][2] : ' ');
	2	-	Inner loop completes for j = 0 to COL-1.	
	2	-	cout << "|"<<endl	
	2	-	cout << " _ _ _ _ _ _ _" << endl;	_ _ _ _ _ _ _"
Outer LOOP	….	…	Continue outer loop until i < ROW condition is false.	
Final LOOP	-	-	Function grid() execution completes.	

Function#4

`bool insertInColumn(int col, char player)`

Iteration	Input 'col'	Initial 'arr'	Action	Final 'arr'	Output	Return Value
1	3	Empty	Check `col` validity (`0 <= col < 7`)			
			Loop from bottom row to top (`i = 5 to 0`)			
			Check if `arr[i][3]` is empty (`'\0'`)	arr[5][3] = 'x'		'True'
			Exit loop, update `arr[i][3]`			

			with `player`			
				{'\0', '\0', '\0', 'x', '\0', '\0', '\0'}		
2	2	{'\0', '\0', '\0', 'x', '\0', '\0', '\0'}	Check `col` validity (`0 <= col < 7`)			
			Loop from bottom row to top (`i = 5 to 0`)			
			Check if `arr[i][2]` is empty (`'\0'`)	arr[5][2] = 'o'		'True'
			Exit loop, update `arr[i][2]` with `player`			

Function#5:

`bool gameover(char player);`

Iteration	i	j	Conditional Check	Action	Retrurn
Initial	-	-	-	Function gameover(player) is called.	-
	-	-	-	Check horizontal win for player.	-
	0	0	arr[0][0] == player && arr[0][1] == player && ...	No match, continue checking.	-
	-	-		Check vertical win for player.	-
	0	0	arr[0][0] == player && arr[1][0] == player && ...	No match, continue checking.	-
	-	-		Check diagonal (bottom-left to top-right).	-

	3	0	arr[3][0] == player && arr[2][1] == player && ...	No match, continue checking.	-
	-	-		Check diagonal (top-left to bottom-right).	-
	0	0	arr[0][0] == player && arr[1][1] == player && ...	No match, continue checking.	-
Final				All checks completed, no winning condition found.	false

Explanation:

- **Initialization**: The function begins execution.
- **Horizontal Check**: Iterates through each row (i) and checks sets of four consecutive columns (j to j+3) to see if they contain all player markers.
- **Vertical Check**: Iterates through each column (j) and checks sets of four consecutive rows (i to i+3) to see if they contain all player markers.
- **Diagonal Checks**:
  - **Bottom-left to top-right diagonal**: Starts from rows where i >= 3 and checks four consecutive diagonal elements.
  - **Top-left to bottom-right diagonal**: Starts from rows where i <= ROW - 4 and checks four consecutive diagonal elements.
- **Return**: Returns true if any of the checks find four consecutive player markers in a line (horizontal, vertical, or diagonal), otherwise returns false.

This compact table captures the essential steps and conditions checked during the execution of the gameover function, making it easy to follow the logic and outcomes. Adjust the values and conditions based on specific test scenarios or actual gameplay data.

Function#6:

draw();

Iteration	j	Condition checked	Action	Return Value

Initial	-	-	Function draw() is called.	-
	0	arr[0][0] == '\0	Check if top row column is empty.	false
Final	-	j < COL (where COL is the number of columns)	All columns checked, none empty.	True

- **Initialization**: The function begins execution.
- **Column Check**: Iterates through each column (j) of the top row (arr[0][j]) to check if it is empty ('\0' indicates empty for this example).
- **Return**: Returns false if any column in the top row is empty. Otherwise, returns true if all columns in the top row are filled.

This table summarizes the key steps and conditions checked during the execution of the draw() function. Adjust the conditions and values based on your specific implementation and test scenarios.

Step-by-Step Execution

`main()` Function:

- Calls `mannual()` and `instruction()` to display welcome messages and get player input for instructions/rules.
- Initializes variables and clears the game grid (`arr`).
- Prompts players to enter their names.
- Enters the main game loop:
  - Displays the game grid.
  - Asks Player 1 (using 'X') to input a column to place their marker.
  - Checks if the input is valid using `insertInColumn()`.
  - Checks if Player 1 has won using `game over ()`.
  - Checks if the game is a draw using `draw()`.

    - Alternates to Player 2 (using 'O') and repeats the process.
- Ends the game when a player wins or when it's a draw.
- Prints the winner or declares a draw.

Example Gameplay:

Assume Players:

- Player 1: Ali (X)
- Player 2: Ahsan (O)

Initial Setup:

- Displays welcome message and instructions.
- Prompts for player names.

Game Start:

- Grid is displayed.
- Ali (Player 1) inputs column 3.
- Grid updates with 'X' in column 3.
- Ahsan (Player 2) inputs column 4.
- Grid updates with 'O' in column 4.
- Players continue alternating turns until:
    - Ali connects four 'X' horizontally.
    - Ahsan connects four 'O' vertically.
    - All columns are filled without a winner (draw).

End of Game:

- Displays the winner (Awins!) or declares a draw.s