# Connect – 3 (Project)

## The Connect-3 game code I provided earlier has the following features:

1. Two-player game: The game can be played by two players taking turns to drop checkers onto the game board.
2. Fixed game board size: The game board is a fixed size of 7 by 7.
3. Input validation: The code performs input validation to ensure that the player's move is valid. It checks if the specified column is not full and prompts the player to enter a valid column number if the input is invalid.
4. Winner detection: The code detects if a player has placed three of their checkers next to each other vertically, horizontally, or diagonally and declares them the winner.
5. Tie detection: The code detects if the game board is full and no player has three consecutive checkers and declares the game a tie.
6. Console-based interface: The game is played in the console using the text-based output and keyboard input.
7. Simple game logic: The game logic is relatively simple, making it easy to understand and modify.

## A brief explanation of each function

1. init_board() - This function initializes the game board by setting all cells to 0, which represents an empty cell.

2. display_board() - This function displays the current state of the game board on the console.

3. is_valid_move(col) - This function checks if the move entered by the player is valid. It returns 1 if the column is not full and 0 otherwise.

4. drop_checker(col, checker) - This function drops a checker of the given type (player1 or player2) into the given column. It returns the row index where the checker was dropped.

5. check_winner(row, col, checker) - This function checks if the given player has won the game by placing three checkers in a row (horizontally, vertically, or diagonally) starting from the given row and column. It returns 1 if there is a winner and 0 otherwise.

6. is_tie() - This function checks if the game has ended in a tie by scanning the entire board for any empty cells. It returns 1 if there is a tie and 0 otherwise.

7. main() - This is the main function that controls the flow of the game. It initializes the board, displays it, and starts a loop where players take turns entering their moves. It checks for a winner or ties after each move and displays the appropriate message before ending the game.

## How each function works:

1. init_board(): This function uses nested loops to initialize the game board to all zeros. The game board is represented by a two-dimensional array where each element is an integer that can have a value of 0, 1, or 2, representing an empty cell, a cell occupied by player 1, or a cell occupied by player 2.

2. display_board(): This function uses nested loops to print the game board to the console. It prints a horizontal line of underscores at the top to separate the column numbers from the board, then prints the column numbers in sequence. It then iterates through each row and column of the board and prints the corresponding cell value as a character (' ', 'O', or 'X') with a vertical bar separating each column.

3. is_valid_move(col): This function checks if a move is valid by checking if the specified column is full (i.e. if the highest row in the column has already been filled with a checker). It returns 1 if the column is not full and 0 otherwise.

4. drop_checker(col, checker): This function drops a checker into the specified column, starting from the bottom row and moving up until it finds an empty cell. It updates the corresponding cell in the game board to the player's checker value (1 or 2) and returns the row index where the checker was dropped.

5. check_winner(row, col, checker): This function checks if the specified player has won the game by placing three checkers in a row (horizontally, vertically, or diagonally) starting from the given row and column. It uses nested loops to check for three consecutive checkers in each of the eight possible directions (horizontal, vertical, diagonal up, diagonal down, and their opposites). If it finds three consecutive checkers of the same player, it returns 1 to indicate a winner; otherwise, it returns 0.

6. is_tie(): This function checks if the game has ended in a tie by scanning the entire board for empty cells. It uses nested loops to check each cell of the game board. If it finds any empty cell, it returns 0 to indicate that the game is not over; otherwise, it returns 1 to indicate a tie.

7. main(): This is the main function that controls the flow of the game. It calls init_board() to initialize the game board, display_board() to show the initial state of the board, and then starts a loop that alternates between the two players. For each turn, it prompts the player to enter a column number, checks if the move is valid using is_valid_move(), drops a checker using drop_checker(), checks if the player has won using check_winner(), checks if the game has ended in a tie using is_tie(), and then updates the current player. If a winner or tie is detected, it displays the appropriate message and exits the loop.

## Here are some possible questions that could be asked regarding this code:

1. How does the game board work?
2. How is a player's move validated?
3. How is a checker dropped onto the game board?

4. How is a winner determined?
5. How is a tie detected?
6. How is the game loop controlled?
7. Can the game board be resized? If so, how?
8. Can the game be played by more than two players? If so, how would the code need to be modified?
9. How would you modify the code to allow a player to remove a checker from the game board?
10. How would you modify the code to add a feature that allows players to choose the colour of their checkers?

## Here are brief answers to the questions:

1. The game board is represented by a 2D array of integers, where each integer represents an empty cell (0), a cell occupied by player 1 (1), or a cell occupied by player 2 (2).

2. A player's move is validated by checking if the specified column is not full (i.e. if the highest row in the column has not already been filled with a checker).

3. A checker is dropped onto the game board by iterating through the rows of the specified column from bottom to top until an empty cell is found. The corresponding cell in the game board is then updated with the player's checker value (1 or 2).

4. A winner is determined by checking each of the eight possible directions (horizontal, vertical, diagonal up, diagonal down, and their opposites) for three consecutive checkers of the same player.

5. A tie is detected by scanning the entire game board for empty cells. If no empty cells are found, the game is declared a tie.

6. The game loop is controlled using a while loop that alternates between the two players until a winner or tie is detected.

7. The game board cannot be resized in this implementation, as it is a fixed size of 7 by 7.

8. The code would need to be modified to allow for more than two players. One way to do this would be to modify the game board to accommodate more players and update the checker values accordingly.

9. To allow a player to remove a checker from the game board, you would need to modify the **drop_checker()** function to remove the checker from the game board instead of adding it. You would also need to add input validation to ensure that the player is only removing their checkers.

10. To allow players to choose the colour of their checkers, you would need to modify the **display_board()** function to print the checkers in the desired colour, based on each player's input. You would also need to modify the **drop_checker()** function to update the game board with the correct colour value.