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. **displayBoard():** This function displays the game board to the console. It loops through each row and column of the board and prints the corresponding character to the console.
2. 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.
3. **makeMove():** This function takes the player's input (a column number) and places their piece in the first available slot in that column. It starts at the bottom of the column and checks each row until it finds an empty space. Once it finds an empty space, it places the player's piece in that slot.
4. **checkWin():** This function checks if a player has won the game. It loops through each row, column, and diagonal of the board and counts how many pieces of the player's color are in a row. If there are three pieces in a row, the function returns 1 (indicating a win). If there are no wins found, the function returns 0.
5. **isTie():** 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.
6. **main():** This is the main function of the program. It initializes the game board, loops through turns, displays the board to the console, gets player input, makes a move on the board, checks for a win, and switches the active player. The loop continues until a player wins, at which point the game ends and the function returns 0.

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

1. How does the makeMove() function handle invalid input from the player (e.g. input that is not a valid column number or a column that is already full)?
2. What happens if the game ends in a tie (i.e. if the board is completely filled and neither player has won)?

3. How would you modify the checkWin() function to allow for larger game boards (e.g. 8x8 or 10x10)?
4. How would you modify the displayBoard() function to use ASCII art instead of simple characters to represent the game pieces?
5. How would you modify the main() function to allow for the players to choose their own game pieces (e.g. X or O, or even different symbols)?

Here are brief answers to the questions:

1. The current implementation of makeMove() assumes that the player input is a valid column number (between 0 and 6), and does not handle invalid input. To handle invalid input, have add an other function that checks whether or not the column is full or not.
2. The isTie() function returns true and 'It is a Tie' is prompted.
3. Just change the value of the constant BOARD_SIZE to 8 or 10 (Keeping in mind that the it is still a connect - 3 game)
4. To modify the displayBoard() function to use ASCII art, we could replace the simple characters with more complex characters that resemble game pieces. For example, we could use the characters "X" and "O" for the players' pieces, or we could use custom ASCII characters that resemble game pieces (e.g. squares or circles).
5. To modify the main() function to allow for custom game pieces, we could prompt each player to choose their own game piece at the beginning of the game. We could then store these game pieces as variables and use them to display the board and check for wins. For example, we could prompt the players to enter a single character (e.g. "X" or "O") to represent their game piece.