## **Instructions**

**Objective:**

Develop a console-based Tic Tac Toe game where the game board and moves are managed using a singly linked list data structure. This assignment aims to deepen your understanding of linked lists and how they can be applied to manage dynamic data in applications.

**Description:**

You are tasked with creating a two-player Tic Tac Toe game. The game should allow players to take turns placing their marks ('X' or 'O') on a 3x3 grid. The twist is that the game board must be represented using a singly linked list rather than a traditional two-dimensional array.

**Requirements:**

1. **Data Structure Implementation:**
   * **Node Structure:**
     + Each node in your singly linked list should represent a single cell on the Tic Tac Toe board.
     + The node should contain:
       - An identifier for the cell position (e.g., numbers 1 through 9).
       - A value indicating the cell's current state ('X', 'O', or empty).
       - A reference (pointer) to the next node.
   * **Linked List Construction:**
     + Initialize the linked list to represent an empty 3x3 Tic Tac Toe board.
     + Ensure that the nodes are connected in the correct order to represent the board positions.
2. **Game Functionality:**
   * **Display Board:**
     + Create a function to display the current state of the board in a 3x3 grid format.
     + The display should update after each move to reflect the current board state.
   * **Player Moves:**
     + Prompt players alternately to enter their move by selecting an empty cell number (1-9).
     + Validate inputs to ensure the selected cell is within range and not already occupied.
   * **Update Board:**
     + Upon a valid move, update the corresponding node in the linked list with the player's mark.
   * **Win Condition Check:**
     + After each move, check if the current player has won the game.
     + Winning conditions include three of the same marks in a horizontal, vertical, or diagonal row.
   * **Draw Condition:**
     + If all cells are filled and no player has won, declare the game a draw.
   * **Game Replay Option:**
     + After a game concludes, prompt the players to either play again or exit the game.
     + If they choose to play again, reset the linked list to its initial empty state.
3. **Technical Constraints:**
   * **Data Structure Usage:**
     + Do **not** use arrays, array lists, or any other collection classes for storing the board.
     + All board operations must be performed using the singly linked list.
   * **Memory Management:**
     + Ensure that any dynamically allocated memory is properly managed.
     + Free or delete nodes appropriately when they are no longer needed (if applicable in your programming language).
4. **Programming Practices:**
   * **Modular Design:**
     + Divide your code into functions or methods for initialization, display, input handling, updating the board, and checking win conditions.
     + Each function should have a single responsibility.
   * **Code Readability:**
     + Use clear and descriptive variable and function names.
     + Include comments to explain complex sections of code.
   * **Error Handling:**
     + Anticipate and handle potential errors gracefully, such as invalid input or unexpected null references.

**Resources:**

* **Linked List Tutorials:**
  + Review materials on singly linked lists to refresh your understanding.
  + Pay special attention to node insertion, traversal, and updating node values.
* **Sample Code Structures:**
  + While you should write your own code, looking at examples of linked list implementations can provide guidance on best practices.