Kathleen Wims
CSC 229 Data Structures and Algorithms I 92516
Module 2 Assignment - Implement Tic Tac Toe Using a Single Linked List

Program Overview

This Tic-Tac-Toe program implements a console based Tic-Tac-Toe game. The game allows players to place their mark ('X' or 'O') on a 3x3 grid. In this game, the board and moves are managed by a singly linked list. A singly linked list is a dynamic data structure where each element (`Node`) contains data and a reference to the next node. The last node points to null, which indicates the end of the list. In this program, the linked list consists of 9 nodes, which represent the 9 cells of the tic-tac-toe board. The program consists of the following key components:

1. **Node Class**:
    - Each `Node` represents a cell in the Tic-Tac-Toe board.
    - The `Node` stores:
        - `data`: The value indicating the cell's current state (`X`, `O`, or an empty string).
        - `cellPosition`: An identifier for the cell's position on the board (1–9).
        - `next`: A reference (pointer) to the next node.
2. **LinkedList Class**:
    - This class contains all of the game board operations using the singly linked list. Many of the methods involve traversing the list to find a specific `Node` object based on its `cellPosition`.
    - Key methods include:
        - `insertAtEnd()`: Adds a new node at the end of the linked list.
        - `updatePosition()`: Updates the `data` value (`X` or `O`) at a specific cell.
        - `getPositionValue()`: Gets the `data` value stored in a cell based on its position.
        - `displayBoard()`: Displays the current state of the board in a 3x3 grid format with ASCII art for `X` and `O`.
        - `checkWinCondition()`: Checks if the current player has won by evaluating all possible winning conditions.
        - `isBoardFull()`: Checks if the board is full to determine if the game is a draw.
        - `resetBoard()`: Resets the board by setting the `data` value to an empty string for all nodes.

3. **TicTacToe Class**:
    - This class manages the game. It is responsible for player turns, switching between players, prompting the player for input, and managing game logic. It uses the `LinkedList` class to interact with the board and keep track of the current state.

Challenges

While working on this program I encountered some challenges. Initially, the game was working as it was supposed to. I decided that I should check for a draw, so I had to make a mark in every cell. That's when I encountered the error in my code. When I entered "1" to place a mark in the first cell, the game froze. The updated board did not display, nor did the game prompt the next player to enter their move. This problem was only happening when “1” was entered. To determine where the error was, I added debugging statements to trace the flow of the code. These pointed me to the `isBoardFull()` method. I realized that I had forgotten to include the line `current = current.next;` in the while loop. This resulted in the program getting stuck in an infinite loop, checking the same node without moving to the next one. The error only resulted in an infinite loop when the first node’s `data` value was not empty, which is why the error didn’t happen for any of the other numbers. After adding the missing line, the loop properly traversed the linked list, resolving the issue and allowing the game to run smoothly. Another challenge I encountered was coming up with an efficient solution to check the win condition logic. I considered using mathematical calculations to determine the win conditions, but I ended up just using if statements to check each win condition. I created a helper function to make the code easier to read. I also continuously ran into issues accessing the 9th cell. I used code from the `insertAtEnd()` method for many of the game methods and had to go back and change them to be able to include the 9th cell. I did not implement any optional enhancements for this project. If I had more time, I might have tried to add a computer player. I have implemented a computer player for tic-tac-toe in the past using JavaScript, but that code did not translate easily to Java. This project, and the challenges that I encountered while writing this program, helped to reinforce my understanding of singly linked lists.