Data Structures and Algorithms

Lab 1 (5% of final grade)

(100 points total)

All code must be your own and follow the guidelines in lecture notes and textbook. Use proper coding styles including comments and indentation. No code from Internet sources is allowed in this assignment. All code must be in C++ programming language.

1. Implement a recursive algorithm that determines if a string S is a palindrome, that is, it is equal to its reverse. For example, "racecar" is a palindrome.

   Submit screenshots of the results, code files, and a writeup describing your algorithm and the results, in a zip file.

2. Implement Tower of Hanoi algorithm. Then run the implementation with 1, 5, 25, and 125 disks. Record the amount of time it takes for each of these input sizes.

   What do you infer from this experiment? Submit screenshots of the results, code files, and a writeup describing your findings in a zip file.

3. Tic-Tac-Toe is a game played on a three-by-three board. Two players, X and O, alternate in placing their respective marks in the cells of this board, starting with player X. If either player succeeds in getting three of his or her marks in a row, column, or diagonal, then that player wins. Tic-Tac-Toe is a simple example showing how two-dimensional arrays can be used for positional games. Software for more sophisticated positional games, such as checkers, chess, or the popular simulation games, are all based on the same approach using a two-dimensional array for Tic-Tac-Toe.

   Use the Tic-Tac-Toe implementation provided with the lecture to create a text-based two-player Tic-Tac-Toe game. The basic idea is to use a two-dimensional array, *board*, to maintain the game board. Cells in this array store values that indicate if that cell is empty or stores an X or O. That is, *board* is a three-by-three matrix. For example, its middle row consists of the cells *board*[1][0], *board*[1][1], and *board*[1][2]. In our case, we choose to make the cells in the *board* array be integers, with a 0 indicating an empty cell, a 1 indicating an X, and a -1 indicating O. This encoding allows us to have a simple way of testing whether a given board configuration is a win for X or O, namely, if the values of a row, column, or diagonal add up to -3 or 3, respectively.

   Submit a video demo of the results and code files in a zip file.

4. Given two sorted lists, L_1 and L_2, write a program to verify whether the two sorted lists L_1 and L_2, are really the same list of nodes even if they have different starting points. Use various implementation versions of the List data structure:
    a. array
    b. singly linked list
    c. doubly linked list
    d. circularly linked list

    Allow the user to select either of the above List implementations, while taking advantage of the more flexible options to make the program more efficient. Create and test your program with a realistic input. Submit screenshots of the results, code files, and a writeup describing your solutions in a zip file.

5. Given two lists, L_1 and L_2, write a program to concatenate the two lists L_1 and L_2 into a single list L' that contains all the nodes of L_1 followed by all the nodes of L_2. Use various implementation versions of the List data structure:
    a. array
    b. singly linked list
    c. doubly linked list
    d. circularly linked list

    Allow the user to select either of the above List implementations, while taking advantage of the more flexible options to make the program more efficient. Create and test your program with a realistic input. Submit screenshots of the results, code files, and a writeup describing your solutions in a zip file.