IT623: Data Structures – Lab 4 Assignment

1. Implement the program of Stack

Write a program to implement a stack data structure. The stack should be able to perform the following operations:
- Push: Insert an element
- Pop: Remove the top element
- Peek (Top): View the top element without removing it
- Size: Return the number of elements in the stack
- isEmpty: Check whether the stack is empty

Examples:

Input: Push(10), Push(20), Pop(), Peek()

Output: Stack after push: [10,20], After pop: [10], Peek: 10

Input: isEmpty(), Size()

Output: False, 1

2. Infix to Postfix Conversion

Write a program to convert an infix expression to a postfix expression using a stack. Infix expressions are the standard arithmetic notation (e.g., A + B), whereas postfix expressions (Reverse Polish Notation) place operators after operands (e.g., AB+).

Examples:

Input: A + B * C

Output: ABC*+

Input: (A + B) * C

Output: AB+C*

3. Tower of Hanoi

Write a recursive program to solve the Tower of Hanoi problem. The objective of the puzzle is to move all disks from a source rod to a destination rod using an auxiliary rod, following these rules:

1. Only one disk can be moved at a time.
2. A disk can only be placed on top of a larger disk or on an empty rod.

Examples:

Input: 2 disks

Output: Move disk 1 from A to B
Move disk 1 from A → B
Move disk 2 from A → C
Move disk 1 from B → C

Input: 3 disks

Output: Sequence of 7 moves showing the correct transfer

Move disk 1 from A → C
Move disk 2 from A → B
Move disk 1 from C → B
Move disk 3 from A → C
Move disk 1 from B → A
Move disk 2 from B → C
Move disk 1 from A → C

4. Check Balanced Parenthesis

Write a program to check whether a string has balanced parentheses using a stack. A string is considered balanced if every opening bracket has a corresponding closing bracket in the correct order.

Examples:

Input: (A + B) * (C + D)

Output: Balanced

Input: (A + B * (C - D)

Output: Not Balanced

5. Doubly Circular Linked List

Write a program to implement a doubly circular linked list. The list should allow insertion, deletion, and traversal in both forward and backward directions. Each node will have links to both its previous and next nodes, and the last node should link back to the first node to form a circle.

Examples:

Input: Insert 10, Insert 20, Insert 30

Output: Forward: 10 -> 20 -> 30 -> 10, Backward: 30 -> 20 -> 10 -> 30

Input: Delete 20

Output: Forward: 10 -> 30 -> 10, Backward: 30 -> 10 -> 30

6. Find the Middle Element of the Linked List

Write a program to find the middle element of a singly linked list. If the number of elements is even, return the second middle element.

Examples:

Input: 1 -> 2 -> 3 -> 4 -> 5

Output: 3

Input: 10 -> 20 -> 30 -> 40

Output: 30

Instructions:

1. Do not use ChatGPT or other AI tools to generate the code.

2. Write the code yourself to understand the concepts better.

3. Include any references or links you used to complete the assignment at the end of your submission.