National University of Computer and Emerging Sciences

Data Structures (CL2001)

Date: November 13th, 2024

Course Instructor(s)

Alishba Subhani

HACKATHON

Total Time: 120 minutes

Total Marks: 5

Total Questions: 02

Semester: SP-2024

Campus: Karachi

Dept.: AI

- Create one word and one .cpp file.
- Filename must be in the format K23xxxx_K23xxxx
- Submission has to be made by only one member of the group

Student Name	Roll No	Section	Student Signature

Q1. Provide an in-depth solution of the given question. Make sure to include every aspect of your suggested approach and write that in a simple word file. [7.5 Points, 2.5 Weightage]

Scenario: You are tasked with designing a real-time parking lot management system that uses a stack-based structure to manage cars entering and exiting the lot. The parking lot has a single entry/exit point, and cars must be rearranged when a specific car needs to leave. Design an "Optimized Parking Stack" where cars are parked in a stack, and the system retrieves any requested car with minimal moves.

1. **Auxiliary Stack Allowed**: You may use one auxiliary stack for rearrangement but no other extra storage.
2. **Efficient Search**: Aim for search complexity better than O(n) to locate a car quickly within the stack.
3. **Minimize Operations**: Focus on reducing the number of moves needed to retrieve the car, not customer wait time.

Outline the step-by-step logic to efficiently retrieve a requested car while maintaining the stack's order

Q2. Write the code for the following question and submit a .cpp file. [7.5 Points, 2.5 Weightage]

Given a binary search tree, implement a function that finds all paths from the root to a target leaf node where the sum of values along the path equals a given target sum. The structure should support:

- **insert(value)**: Inserts a node into the BST.
- **findPaths(target_sum)**: Finds all root-to-leaf paths where the sum of the node values equals target_sum.

Use a stack to track the current path during traversal, allowing for backtracking when necessary.