**CSE-2212: Design and Analysis of Algorithms-I Lab**

**Practice Lab 8– June 24, 2025**

**Experiment:** In this lab experiment, you will have to implement four algorithms in Java—

1. Fibonacci Numbers with Memoization and Tabulation in two different functions
2. 0/1 Knapsack
3. LCS
4. Rock Climbing

**Problem 1: Fibonacci Numbers**

You will implement the Fibonacci sequence using two approaches: **Memoization** and **Tabulation**.

**Methods to Implement:**

1. **fibMemo(int n, HashMap<Integer, Integer> memo)**: Computes Fibonacci numbers using a top-down memoization approach.
2. **fibTab(int n)**: Computes Fibonacci numbers using a bottom-up tabulation approach.

**Example Input:**

fibMemo(10)

fibTab(10)

**Expected Output:**

fibMemo: 55

fibTab: 55

**Problem 2: Knapsack Problem**

You will solve the 0/1 Knapsack problem using a dynamic programming approach to maximize the value of items in the knapsack without exceeding its capacity.

**Method to Implement:**

**knapsack(int[] weights, int[] values, int capacity)**: Returns the maximum value that can fit in the knapsack of given capacity.

**Example Input:**

**Number of Items, N = 3**

weights = {1, 2, 3}

values = {10, 15, 40}

capacity = 4

**Expected Output:**

50

**Problem 3: Longest Common Subsequence (LCS)**

You will find the length of the Longest Common Subsequence (LCS) between two strings using dynamic programming.

**Method to Implement:**

**lcs(String s1, String s2)**: Returns the length of the longest common subsequence between two strings.

**Example Input:**

s1 = "AGGTAB"

s2 = "GXTXAYB"

**Expected Output:**

4 (The LCS is "GTAB")

**Problem 4: Rock Climbing Problem**

You will solve the rock climbing problem, where a climber has to reach the top of a wall with certain energy levels on each move. The goal is to maximize the energy collected.

**Method to Implement:**

**rockClimbing(int[][] wall)**: Given a 2D array representing energy values, calculate the maximum energy collected while climbing from bottom to top.

**Example Input:**

wall = {

{1, 2, 3},

{4, 5, 6},

{7, 8, 9}

}

**Expected Output:**

15 (Path: 3 → 6 → 9)

**Bonus: 0/1 Knapsack -** [**https://codeforces.com/contest/1207/problem/C**](https://codeforces.com/contest/1207/problem/C)