****

**Green University of Bangladesh**

**Department of Computer Science and Engineering(CSE)**

**Faculty of Sciences and Engineering**

**Semester: (Spring, Year:2023), B.Sc. in CSE (Day)**

**Report-04**

**Course Title: Algorithm Lab**

**Course Code: CSE 206 Section: DA**

**Student Details**

| **Name** | | **ID** |
| --- | --- | --- |
| **1.** | . Rani Mia | 213902058 |

**Submission Date : 09 June 2023**

**Course Teacher’s Name : Sultanul Islam Ovi**

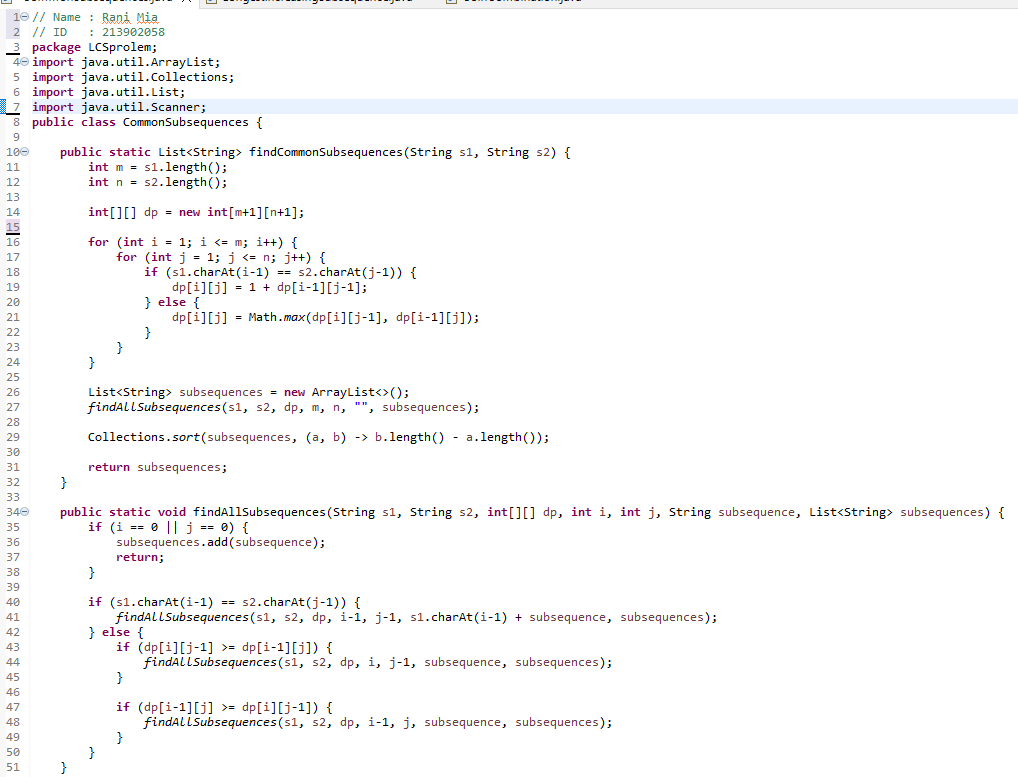
**[For Teachers use only: [Don’t Write Anything inside this box]**

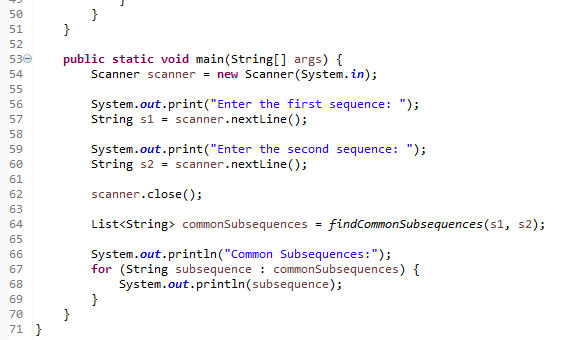
| **Assignment Status**  **Marks: ………………………………… Signature:.....................**  **Comments:.............................................. Date:..............................** |
| --- |

**Question 01:**

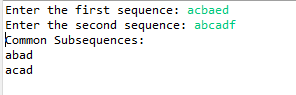
Print all the common subsequences according to the descending order of the lengths for two given sequences. (LCS)

**Input:**

****



**Output:**



**Explanation:**

Okay, I will explain this code step by step.

The **findCommonSubsequences** method takes two strings s1 and s2 as inputs and returns a list of common subsequences.

The lengths of the input strings are stored in variables m and n, respectively.

A 2D array dp of size (m+1) x (n+1) is created to build the dynamic programming table. The dynamic programming table is built using a nested loop that iterates through each character of s1 and s2. If the characters at the current positions in s1 and s2 are the same, the value in the dp table is updated by incrementing the value at the previous diagonal position by 1. If the characters are different, the value in the dp table is updated by taking the maximum of the value above or the value to the left.

After building the dp table, an empty list of subsequences is created to store the common subsequences.

The findAllSubsequences method is called with the input strings, dp table, and other parameters to find all the common subsequences recursively.

In the findAllSubsequences method, if either i or j becomes 0 (indicating the end of a string), the current subsequence is added to the subsequences list.

If the characters at the current positions in s1 and s2 are the same, the findAllSubsequences method is recursively called with the updated positions and the current character appended to the subsequence.

If the characters are different, two recursive calls are made: one by decrementing i and keeping j the same, and the other by decrementing j and keeping i the same. This explores both possibilities of excluding a character from either string.

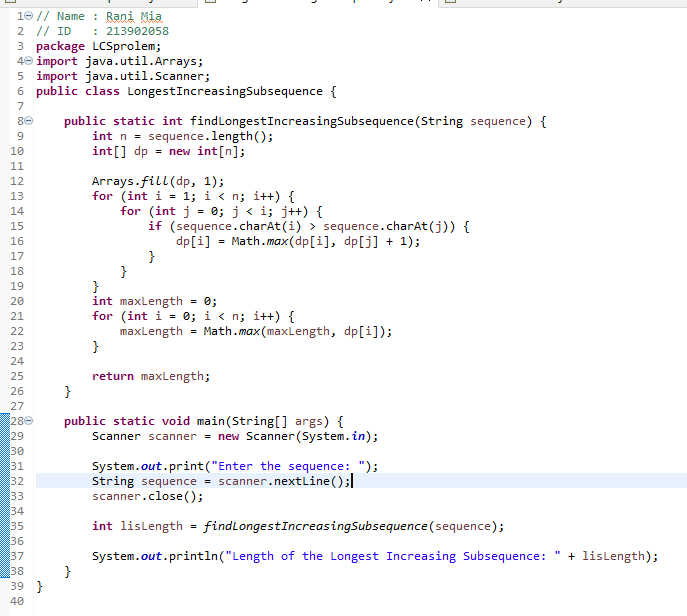
The subsequences list is sorted in descending order of lengths using a comparator.

The main method demonstrates an example with two strings: "abcded" and "abcadf". The findCommonSubsequences method is called with these strings, and the resulting common subsequences are printed to the console

**Question 02:**

Implement Longest increasing subsequence problem using DP technique. (LIS)

**Input:**

****

**Output:**

****

**Explanation:**

The findLongestIncreasingSubsequence method takes a sequence as input and returns the length of the LIS.

The length of the input sequence is stored in the variable n.

An integer array dp of size n is created to store the LIS values for each position.

The dp array is initialized with 1s, as every element forms a subsequence of length 1.

The LIS is computed using dynamic programming. The outer loop iterates from the second element to the last element of the sequence.

The inner loop iterates from the first element to the current element of the outer loop.

If the current element at index i is greater than the element at index j, it means it can be included in the LIS. The dp value at index i is updated to the maximum of its current value or the LIS value at index j plus 1.

After the completion of the dynamic programming loop, the maximum value in the dp array is found, which represents the length of the LIS.

The length of the LIS is returned from the method.

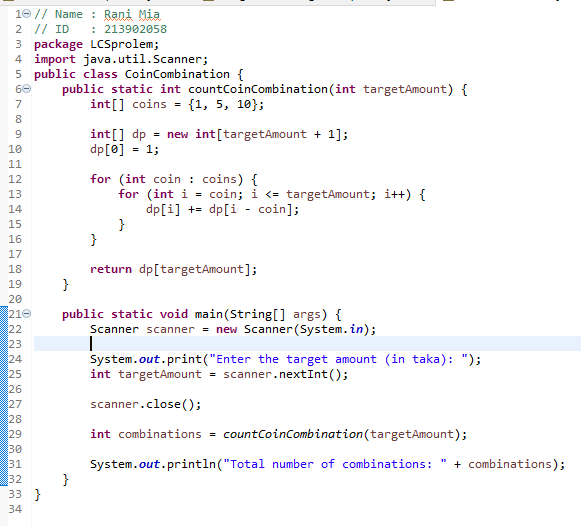
In the main method, the user is prompted to enter the sequence using the Scanner class.

The findLongestIncreasingSubsequence method is called with the user-provided sequence, and the result is stored in the lisLength variable.

**Question 03:**

Given a list of coins i.e 1 taka, 5 taka and 10 taka, can you determine the total number of combinations of the coins in the given list to make up the number N taka?

**Input:**

****

**Output:**

****

**Explanation:**

The countCoinCombinations method takes the target amount as an input and returns the total number of combinations.

The available coin denominations are stored in the coins array, which contains {1, 5, 10}.

An array dp of size targetAmount + 1 is created to store the number of combinations for each amount from 0 to the target amount.

The base case dp[0] = 1 is set, as there is only one way to make an amount of 0.

The dynamic programming loop iterates through each coin denomination. For each coin, it starts from the coin value and goes up to the target amount.

At each iteration, it updates the dp array by adding the number of combinations for the current amount i with the number of combinations for the amount i - coin. This accounts for the combinations using the current coin denomination.

Finally, the method returns the value stored in dp[targetAmount], which represents the total number of combinations to make up the target amount.

In the main method, the user is prompted to enter the target amount using the Scanner class.

The countCoinCombinations method is called with the user-provided target amount, and the result is stored in the combinations variable.