**This project has been done as a part of the coursework ITCS-6114/8114: Algorithms and Data Structures.**

**Program and Data Structure Design:**

Given project has been written as 3 separate programs. The programs are written in Java. Following are the names of the Java class files for each functionality:

1. **NormalizedEditDistance** – This program calculates the normalized edit distance between two input sequences. The program takes two sequences as input from the file path and filenames mentioned by the user. The extension .txt need not be provided by the user. Empty sequences will be handled as invalid input sequence and the edit distance will not be computed. The function to calculate the edit distance does the computation of the table with only two rows in memory. The data structure used is an array to hold the values of computation. The rules for computing the values are as per the project document given. The last value in the array is the number of deletions. Then based on the formula the normalized edit distance is calculated and displayed as output.
2. **LongestCommonSubsequence** – This program calculates the longest common Subsequence between two sequences. This method stores the entire table in memory. This program computes the entire table to find the longest common subsequence based on the rules given and stores the table in memory. The data structure used is an array to hold the computed values and a stack to hold the strings when there is a match. It’s an m x n array where m and n are the length of the input sequences. The table is traversed in a bottom up fashion and the values are popped into a stack where both the sequences match. The traversal in the array (table) happens based on the rules provided in the document.
3. **LinearRecursiveLCS** – This program calculates the LCS between two given sequences using linear memory recursion method. This method uses recursion to compute the sequence and doesn’t store the entire table in memory. Using only two rows the forward middle row and reverse middle row are computed. The vertical and horizontal splits are calculated based on the rules mentioned in the project document. Both the forward and reverse middle row is calculated using the same function . Only the input sequences are reversed for reverse middle row calculation. The base cases of recursion handles the function of appending the characters when there is a match. The output sequence is finally returned to the calling function. The data structure used here is an array with two rows to compute the forward and reverse middle rows.

**Programming language used:** Java

**Compiler used:** Java Programming Language Compiler (javac), included in the Java Development Kit

**Summary of what works and doesn’t work in the program:**

1. The assumption for all the programs is that both the input strings should not be empty. The program doesn’t handle even if one of the input sequence is valid.
2. When the input sequence has trailing and leading spaces it will be trimmed so that the computation doesn’t throw an error.
3. The assumption is that input strings will not have whitespaces between them and it is a continuous sequence. The program may not return expected output if input sequences have whitespaces between them.
4. If the input sequence is more than a single line then the program appends all of them into a single string and does the computation.
5. The program handles special characters in sequences.