KARIYA RAJ 21162101011

ALGORITHM ANALYSIS AND DESIGN

PRACTICAL -8

A subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements. Longest common subsequence (LCS) of 2 sequences is a subsequence, with maximal length, which is common to both the sequences.

Given two sequences of characters, P = M, N, O, M; and Q = M, L, N, O, M, find any one longest common subsequence.

In case multiple solutions exist, print any of them. It is guaranteed that at least one non-empty common subsequence will exist.

CODE:

```
package Practice;
import java.util.Scanner;
public class LCS {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
String a = sc.next();
String b = sc.next();

String lcs = findLCS(a, b);
System.out.println("Longest Common Subsequence: " + lcs);
}

private static String findLCS(String a, String b) {
int a_len = a.length();
int b_len = b.length();

int[][] memo = new int[a_len + 1][b_len + 1];

for (int i = 1; i <= a_len; i++) {
```

```
for (int j = 1; j <= b_len; j++) {
if (a.charAt(i - 1) == b.charAt(j - 1)) {
memo[i][j] = 1 + memo[i - 1][j - 1];
} else {
memo[i][j] = Math.max(memo[i - 1][j], memo[i][j - 1]);
}
}
}

int lcsLength = memo[a_len][b_len];
char[] lcsChars = new char[lcsLength];
int i = a_len, j = b_len, index = lcsLength - 1;

while (i > 0 && j > 0) {
if (a.charAt(i - 1) == b.charAt(j - 1)) {
lcsChars[index] = a.charAt(i - 1);
i--;
j--;
index--;
} else if (memo[i - 1][j] > memo[i][j - 1]) {
i--;
} else {
j--;
}
}

return new String(lcsChars);
}
}
```

OUTPUT: