

# Practical 5

## Task A & B

Name: Parth M. Pathak

Section: A6

Batch: B3

Roll Number: 47

## Task A

### Code:

```
import java.util.Arrays;
import java.util.Collections;

public class LCS {

    static class C {
        int val;
        char dir;

        public C(int val, char dir) {
            this.val = val;
            this.dir = dir;
        }
    }

    public C[][] lcsTables(String a, String b) {
        int m = a.length();
        int n = b.length();

        C[][] c = new C[m + 1][n + 1];

        for (int i = 0; i <= m; i++) {
            c[i][0] = new C(0, 'h');
        }
        for (int j = 0; j <= n; j++) {
            c[0][j] = new C(0, 'h');
        }

        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                if (a.charAt(i - 1) == b.charAt(j - 1)) {
                    c[i][j] = new C(c[i - 1][j - 1].val + 1, 'd');
                }
            }
        }
    }
}
```

```

        else {
            if (c[i - 1][j].val >= c[i][j - 1].val) {
                c[i][j] = new C(c[i - 1][j].val, 'u');
            }
            else {
                c[i][j] = new C(c[i][j - 1].val, 'l');
            }
        }
    }
}
return c;
}

public String reconstructLcs(String a, C[][] backtrack) {
    StringBuilder lcsChars = new StringBuilder();
    int i = a.length();
    int j = backtrack[0].length - 1;

    while (i > 0 && j > 0) {
        char direction = backtrack[i][j].dir;
        if (direction == 'd') {
            lcsChars.append(a.charAt(i - 1));
            i--;
            j--;
        }
        else if (direction == 'u') {
            i--;
        }
        else {
            j--;
        }
    }

    return lcsChars.reverse().toString();
}

public static void main(String[] args) {
    LCS lcsFinder = new LCS();
    String a = "AGGTAB";
    String b = "GXTXAYB";

    C[][] tables = lcsFinder.lcsTables(a, b);

    int lcsLength = tables[a.length()][b.length()].val;
    String lcsString = lcsFinder.reconstructLcs(a, tables);

    System.out.println("String 1: " + a);
    System.out.println("String 2: " + b);
    System.out.println("Length of LCS: " + lcsLength);
    System.out.println("LCS: " + lcsString);
}

```

```

    }
}

```

Screenshot:

```

LCS.java
47  int i = a.length();
48  int j = backtrack[0].length - 1;
49
50  while (i > 0 && j > 0) {
51      char direction = backtrack[i][j].dir;
52      if (direction == 'd') {
53          lcsChars.append(a.charAt(i - 1));
54          i--;
55          j--;
56      } else if (direction == 'u') {
57          i--;
58      } else {
59          j--;
60      }
61  }
62
63  return lcsChars.reverse().toString();
64  }
65
66  public static void main(String[] args) {
67      LCS lcsFinder = new LCS();
68      String a = "AGGTAB";
69      String b = "GXTXAYB";
70
71      C[][] tables = lcsFinder.lcsTables(a, b);
72
73      int lcsLength = tables[a.length()][b.length()].val;
74      String lcsString = lcsFinder.reconstructs(a, tables);
75
76      System.out.println("String 1: " + a);
77      System.out.println("String 2: " + b);
78      System.out.println("Length of LCS: " + lcsLength);
79      System.out.println("LCS: " + lcsString);
80  }
81  }
82

```

Output

```

String 1: AGGTAB
String 2: GXTXAYB
Length of LCS: 4
LCS: GTAB
--- Code Execution Successful ---

```

## Task B

Code:

```

#include <stdio.h>
#include <string.h>

```

```

int max(int a, int b) {
    return (a > b) ? a : b;
}

```

```

int LRS(char* a, char* b) {
    int n = strlen(a);
    int m = strlen(b);

```

```

    int c[n + 1][m + 1];

```

```

    for (int i = 0; i <= n; i++) {
        for (int j = 0; j <= m; j++) {
            c[i][j] = 0;
        }
    }

```

```

    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {

```

```

        if (a[i - 1] == b[j - 1] && i != j) {
            c[i][j] = 1 + c[i - 1][j - 1];
        } else {
            c[i][j] = max(c[i - 1][j], c[i][j - 1]);
        }
    }
}

printf("Matrix:\n");
for (int i = 0; i <= n; i++) {
    for (int j = 0; j <= m; j++) {
        printf("%d ", c[i][j]);
    }
    printf("\n");
}

return c[n][m];
}

int main() {
    char a[] = "AABEBCDD";
    char b[] = "AABEBCDD";

    int result = LRS(a, b);
    printf("LRS length: %d\n", result);

    return 0;
}

```

## Output:

The screenshot shows a C Online Compiler interface with the following components:

- Header:** Programiz C Online Compiler, Programiz PRO >
- File:** main.c
- Code Editor:** Contains the C code for calculating the LRS length.
- Output:** Displays the execution results, including the matrix and the LRS length.
- Buttons:** Run, Share, Clear.

**Code Snippet:**

```

20
21
22 for (int i = 1; i <= n; i++) {
23     for (int j = 1; j <= m; j++) {
24         if (a[i - 1] == b[j - 1] && i != j) {
25             c[i][j] = 1 + c[i - 1][j - 1];
26         } else {
27             c[i][j] = max(c[i - 1][j], c[i][j - 1]);
28         }
29     }
30 }
31
32
33 printf("Matrix:\n");
34 for (int i = 0; i <= n; i++) {
35     for (int j = 0; j <= m; j++) {
36         printf("%d ", c[i][j]);
37     }
38     printf("\n");
39 }
40
41 return c[n][m];
42 }
43
44 int main() {
45     char a[] = "AABEBCDD";
46     char b[] = "AABEBCDD";
47

```

**Output:**

```

Matrix:
0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1 1
0 1 1 1 1 2 2 2 2
0 1 1 1 1 2 2 2 2
0 1 1 1 2 2 2 2 2
0 1 1 2 2 2 2 2 2
0 1 1 2 2 2 2 2 3
0 1 1 2 2 2 2 3 3
LRS length: 3

=== Code Execution Successful ===

```

# LeetCode

Code:

```
class Solution {
    public int longestCommonSubsequence(String text1, String text2) {
        int m = text1.length();
        int n = text2.length();
        int[][] dp = new int[m + 1][n + 1];

        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                if (text1.charAt(i - 1) == text2.charAt(j - 1)) {
                    dp[i][j] = dp[i - 1][j - 1] + 1;
                } else {
                    dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]);
                }
            }
        }

        return dp[m][n];
    }
}
```

Output:

The screenshot displays the LeetCode problem page for "1143. Longest Common Subsequence". The problem description states: "Given two strings text1 and text2, return the length of their longest common subsequence. If there is no common subsequence, return 0." It also provides a definition of a subsequence and an example: "For example, 'ace' is a subsequence of 'abcde'." The input for Example 1 is text1 = "abcde", text2 = "ace", and the output is 3. The explanation states: "The longest common subsequence is 'ace' and its length is 3." The Java solution is shown in the code editor, and the test result is "Accepted" with a runtime of 0 ms. The user's profile "Parth\_Pathak05" is visible in the top right corner.

Testcase

Test Result

Accepted

Runtime: 0 ms

Case 1

Case 2

Case 3

Input

text1 =  
"abcde"

text2 =  
"ace"

Output

3

Expected

3

Contribute a testcase

Testcase

Test Result

Accepted

Runtime: 0 ms

Case 1

Case 2

Case 3

Input

text1 =  
"abc"

text2 =  
"abc"

Output

3

Expected

3

Contribute a testcase

Testcase

Test Result

Accepted

Runtime: 0 ms

Case 1

Case 2

Case 3

Input

text1 =  
"abc"

text2 =  
"def"

Output

0

Expected

0

Contribute a testcase