Practical Number: 5

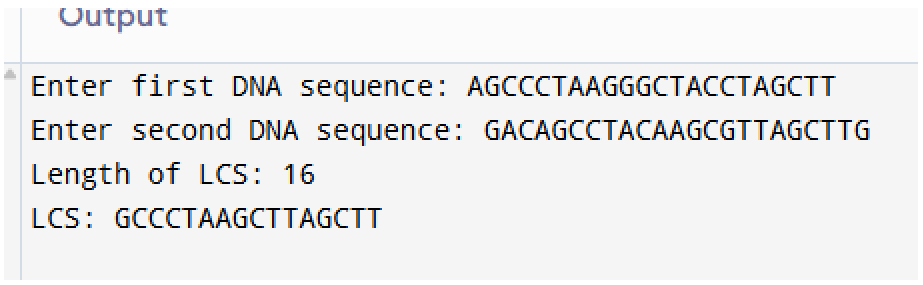
Name: Laghima Kejariwal
Section: A3
Batch: B2
Roll Number: 19
Date: 06-10-2025

Task: 1

Code:

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

#define MAX 100

void findLCS(char *X, char *Y) {
    int m = strlen(X);
    int n = strlen(Y);
    int L[m+1][n+1];

    for (int i = 0; i <= m; i++) {
        for (int j = 0; j <= n; j++) {
            if (i == 0 || j == 0)
                L[i][j] = 0;
            else if (X[i-1] == Y[j-1])
                L[i][j] = L[i-1][j-1] + 1;
            else
                L[i][j] = (L[i-1][j] > L[i][j-1]) ? L[i-1][j] : L[i][j-1];
        }
    }

    int index = L[m][n];

    char lcsStr[index + 1];
    lcsStr[index] = '\0';

    int i = m, j = n;
    while (i > 0 && j > 0) {
        if (X[i-1] == Y[j-1]) {
            lcsStr[index-1] = X[i-1];
            i--;
            j--;
            index--;
        } else if (L[i-1][j] > L[i][j-1]) {
            i--;
        } else {
            j--;
        }
    }

    printf("Length of LCS: %d\n", L[m][n]);
    printf("LCS: %s\n", lcsStr);
}

int main() {

    char X[MAX], Y[MAX];
    printf("Enter first DNA sequence: ");
    scanf("%s", X);
    printf("Enter second DNA sequence: ");
    scanf("%s", Y);

    findLCS(X, Y);

    return 0;
}
```

OutPut:



```
Enter first DNA sequence: AGCCCTAAGGGCTACCTAGCTT
Enter second DNA sequence: GACAGCCTACAAGCGTTAGCTTG
Length of LCS: 16
LCS: GCCCTAAGCTTAGCTT
```

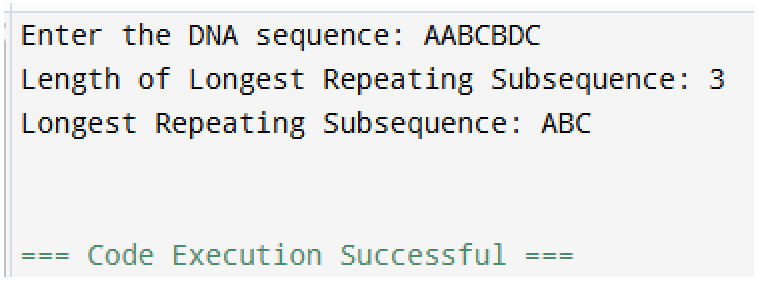
Task: 2

Code:

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

#define MAX 100

void findLRS(char *S) {
    int n = strlen(S);
    int L[n+1][n+1];

    for (int i = 0; i <= n; i++) {
        for (int j = 0; j <= n; j++) {
            if (i == 0 || j == 0)
                L[i][j] = 0;
            else if (S[i-1] == S[j-1] && i != j)
                L[i][j] = L[i-1][j-1] + 1;
            else
                L[i][j] = (L[i-1][j] > L[i][j-1]) ? L[i-1][j] : L[i][j-1];
        }
    }

    int index = L[n][n];

    char lrsStr[index + 1];
    lrsStr[index] = '\0';

    int i = n, j = n;
    while (i > 0 && j > 0) {
        if (S[i-1] == S[j-1] && i != j) {
            lrsStr[index - 1] = S[i - 1];
            i--;
            j--;
            index--;
        } else if (L[i-1][j] > L[i][j-1]) {
            i--;
        } else {
            j--;
        }
    }

    printf("Length of Longest Repeating Subsequence: %d\n", L[n][n]);
    printf("Longest Repeating Subsequence: %s\n", lrsStr);
}
```

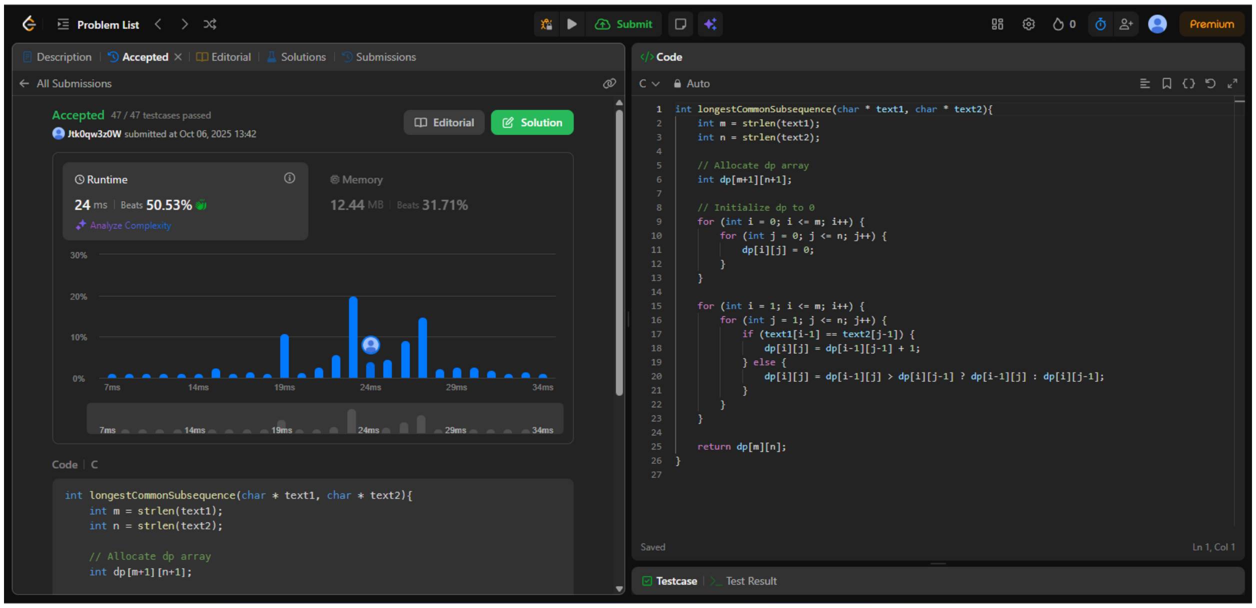
```
int main() {
    char S[MAX];

    printf("Enter the DNA sequence: ");
    scanf("%s", S);

    findLRS(S);

    return 0;
}
```

OutPut:



```
Enter the DNA sequence: AABCBDC
Length of Longest Repeating Subsequence: 3
Longest Repeating Subsequence: ABC



=== Code Execution Successful ===
```

LeetCode:



Accepted 47 / 47 testcases passed
Jtk0qw3z0W submitted at Oct 06, 2025 13:42

Runtime 24 ms | Beats 50.53%

Memory 12.44 MB | Beats 31.71%

Code | C

```
int longestCommonSubsequence(char * text1, char * text2){
    int m = strlen(text1);
    int n = strlen(text2);

    // Allocate dp array
    int dp[m+1][n+1];

    // Initialize dp to 0
    for (int i = 0; i <= m; i++) {
        for (int j = 0; j <= n; j++) {
            dp[i][j] = 0;
        }
    }

    for (int i = 1; i <= m; i++) {
        for (int j = 1; j <= n; j++) {
            if (text1[i-1] == text2[j-1]) {
                dp[i][j] = dp[i-1][j-1] + 1;
            } else {
                dp[i][j] = dp[i-1][j] > dp[i][j-1] ? dp[i-1][j] : dp[i][j-1];
            }
        }
    }

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