DAA PRACTICLE 5

NAME:ASHLESHA LANGDE

SECTION:A8_B2_18

TASK:1

```
#include <stdio.h>

#include <string.h>


#define MAX 100


// Build LCS dynamic programming tables

void lcs(char x[], char y[], int m, int n, int c[MAX][MAX], int
b[MAX][MAX]) {

    for (int i = 0; i <= m; i++) c[i][0] = 0;

    for (int j = 0; j <= n; j++) c[0][j] = 0;


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

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

            if (x[i - 1] == y[j - 1]) {

                c[i][j] = c[i - 1][j - 1] + 1;

                b[i][j] = 1;  // Diagonal ↖ (match)

            } else if (c[i - 1][j] >= c[i][j - 1]) {

                c[i][j] = c[i - 1][j];
```

```
            b[i][j] = 2;  // Up ↑
        } else {
            c[i][j] = c[i][j - 1];
            b[i][j] = 3;  // Left ←
        }
    }
  }

  // Print only final LCS length
  printf("Length of LCS: %d\n", c[m][n]);
}

// Print the LCS using the direction matrix
void printlcs(int b[MAX][MAX], char x[], int i, int j) {
  if (i == 0 || j == 0) return;

  if (b[i][j] == 1) {
    printlcs(b, x, i - 1, j - 1);
    printf("%c", x[i - 1]);
  } else if (b[i][j] == 2) {
    printlcs(b, x, i - 1, j);
  } else {
```

```
        printlcs(b, x, i, j - 1);
    }
}

int main() {
    char x[] = "AGCCCTAAGGGCTACCTAGCTT";
    char y[] = "GACAGCCTACAAGCGTTAGCTTG";

    int m = strlen(x);
    int n = strlen(y);

    int c[MAX][MAX], b[MAX][MAX];

    // Compute LCS
    lcs(x, y, m, n, c, b);

    // Print LCS
    printf("Longest Common Subsequence: ");
    printlcs(b, x, m, n);
    printf("\n");

    return 0;
```

}

```
Length of LCS: 16
Longest Common Subsequence: AGCCCAAGGTTAGCTT


...Program finished with exit code 0
Press ENTER to exit console.
```

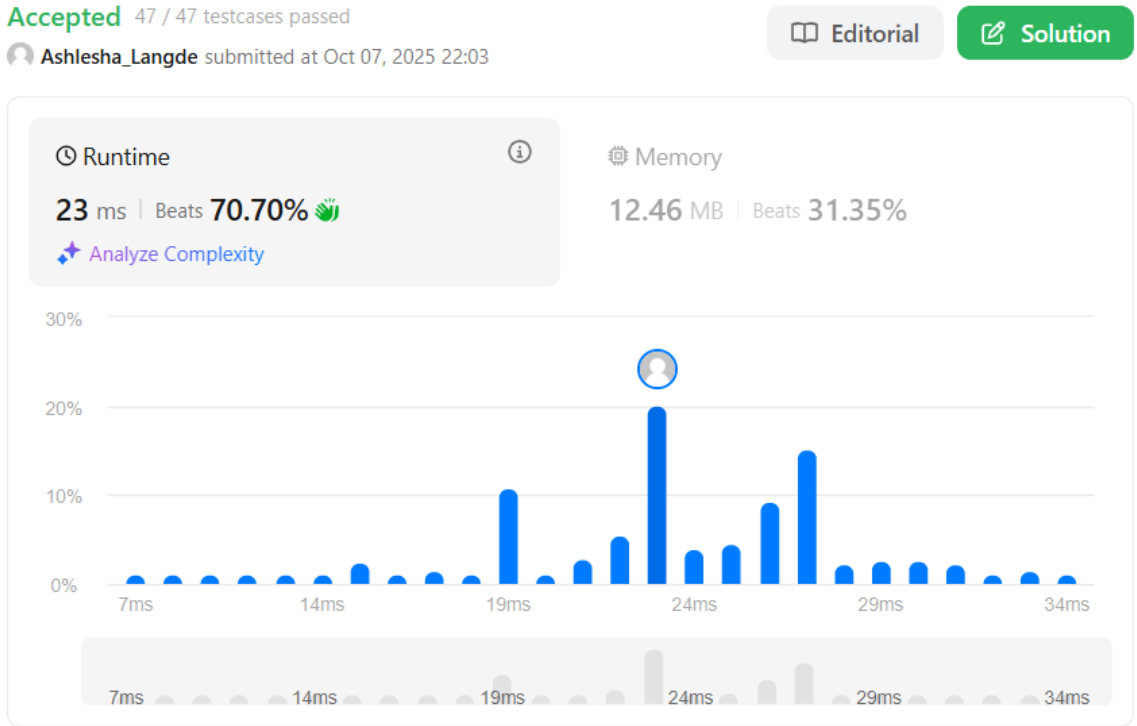
TASK 2:

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

void lrs(char *str) {
    int n = strlen(str);

    // Allocate memory for dp table dynamically
    int **dp = (int **)malloc((n+1) * sizeof(int *));
    for (int i = 0; i <= n; i++) {
        dp[i] = (int *)malloc((n+1) * sizeof(int));
        for (int j = 0; j <= n; j++) {
            dp[i][j] = 0;
        }
    }

    // Fill dp table for Longest Repeating Subsequence
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            if (str[i-1] == str[j-1] && i != j)
                dp[i][j] = 1 + dp[i-1][j-1];
            else
                dp[i][j] = (dp[i-1][j] > dp[i][j-1]) ? dp[i-1][j] : dp[i][j-1];
        }
    }

    printf("The length of the longest repeating subsequence is: %d\n", dp[n][n]);

    // Reconstruct the longest repeating subsequence
```

```
    int i = n, j = n;
    char res[n+1];
    int index = 0;

    while (i > 0 && j > 0) {
        if (str[i-1] == str[j-1] && i != j) {
            res[index++] = str[i-1];
            i--;
            j--;
        } else if (dp[i-1][j] > dp[i][j-1]) {
            i--;
        } else {
            j--;
        }
    }
    res[index] = '\0';

    // Reverse the result since we built it backwards
    for (int k = 0; k < index / 2; k++) {
        char temp = res[k];
        res[k] = res[index - k - 1];
        res[index - k - 1] = temp;
    }

    printf("The longest repeating subsequence is: %s\n", res);

    // Free dynamically allocated memory
    for (int i = 0; i <= n; i++) {
        free(dp[i]);
    }
    free(dp);
}

int main() {
    char str[] = "aabb";
    lrs(str);
    return 0;
}
```

```
The length of the longest repeating subsequence is: 2
The longest repeating subsequence is: ab


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

Leetcode:



Code | C

C Auto

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

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

    // Initialize first row and column to 0
    for (int i = 0; i <= m; i++) {
        for (int j = 0; j <= n; j++) {
            if (i == 0 || j == 0) {
                dp[i][j] = 0;
            } else 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];
}
```