DAA Lab Pract5

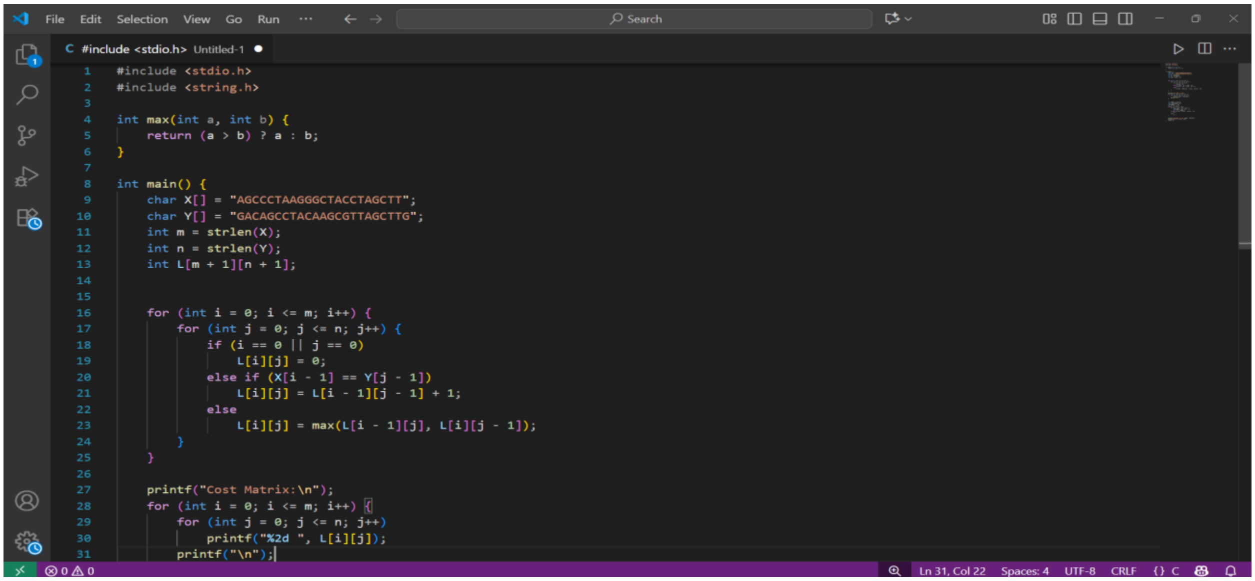
Name: Anshika Tiwari

Roll No : A7_B1_14

Aim: Implement Longest Common Subsequence (LCS) algorithm to find the length and LCS for DNA sequences.

Task 1 :

Code:



File Edit Selection View Go Run ···
Search
C #include <stdio.h> Untitled-1
⊗ 0 ⚠ 0
Ln 31, Col 22 Spaces: 4 UTF-8 CRLF {} C

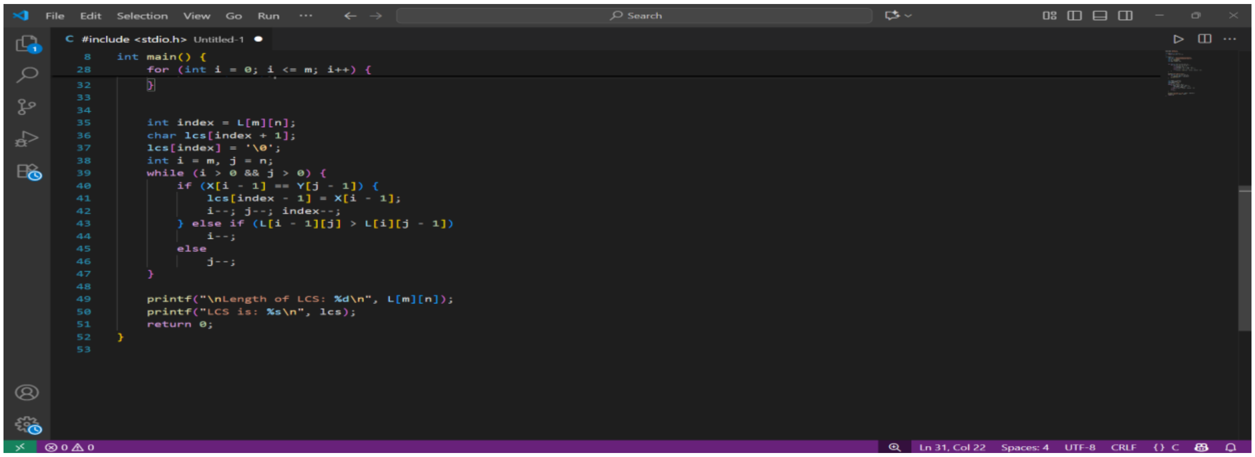
```c
#include <stdio.h>
#include <string.h>

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

int main() {
    char X[] = "AGCCCTAAGGGCTACCTAGCTT";
    char Y[] = "GACAGCCTACAAGCGTTAGCTTG";
    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] = max(L[i - 1][j], L[i][j - 1]);
        }
    }

    printf("Cost Matrix:\n");
    for (int i = 0; i <= m; i++) {
        for (int j = 0; j <= n; j++)
            printf("%2d ", L[i][j]);
        printf("\n");
```



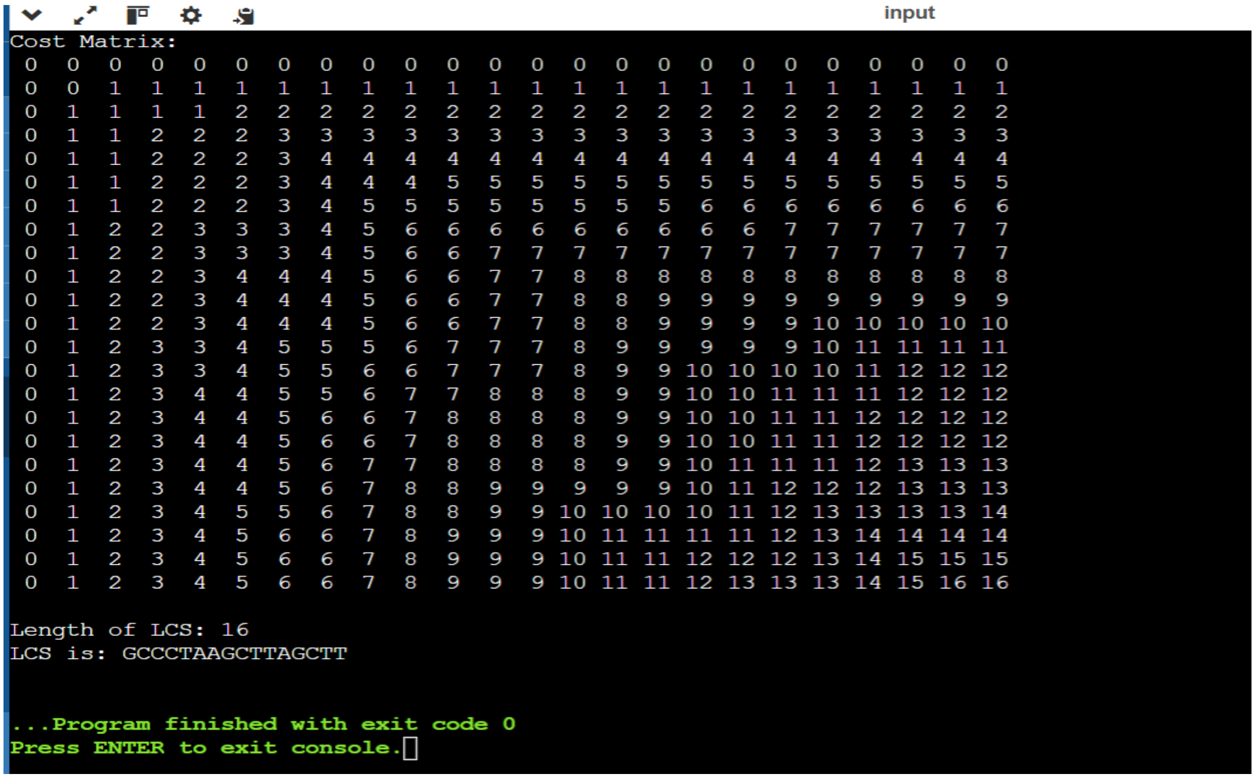
File Edit Selection View Go Run ···
Search
C #include <stdio.h> Untitled-1
⊗ 0 ⚠ 0
Ln 31, Col 22 Spaces: 4 UTF-8 CRLF {} C

```c
int main() {
    for (int i = 0; i <= m; i++) {
    }


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

    printf("\nLength of LCS: %d\n", L[m][n]);
    printf("LCS is: %s\n", lcs);
    return 0;
}
```

Output:



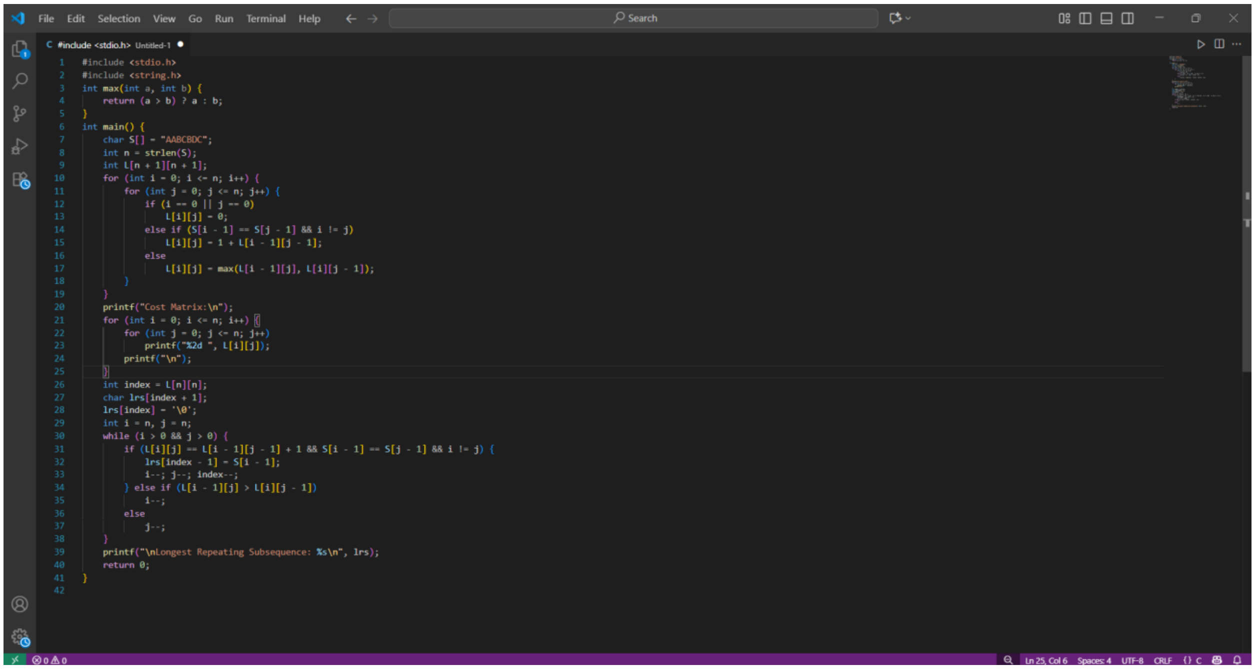
```
Cost Matrix:
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 
 0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 
 0  1  1  1  1  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2 
 0  1  1  2  2  2  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3 
 0  1  1  2  2  2  3  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4 
 0  1  1  2  2  2  3  4  4  4  5  5  5  5  5  5  5  5  5  5  5  5  5  5 
 0  1  1  2  2  2  3  4  5  5  5  5  5  5  5  5  6  6  6  6  6  6  6  6 
 0  1  2  2  3  3  3  4  5  6  6  6  6  6  6  6  6  6  7  7  7  7  7  7 
 0  1  2  2  3  3  3  4  5  6  6  7  7  7  7  7  7  7  7  7  7  7  7  7 
 0  1  2  2  3  4  4  4  5  6  6  7  7  8  8  8  8  8  8  8  8  8  8  8 
 0  1  2  2  3  4  4  4  5  6  6  7  7  8  8  9  9  9  9  9  9  9  9  9 
 0  1  2  2  3  4  4  4  5  6  6  7  7  8  8  9  9  9  9 10 10 10 10 10 
 0  1  2  3  3  4  5  5  5  6  7  7  7  8  9  9  9  9  9 10 11 11 11 11 
 0  1  2  3  3  4  5  5  6  6  7  7  7  8  9  9 10 10 10 10 11 12 12 12 
 0  1  2  3  4  4  5  5  6  7  7  8  8  8  9  9 10 10 11 11 11 12 12 12 
 0  1  2  3  4  4  5  6  6  7  8  8  8  8  9  9 10 10 11 11 12 12 12 12 
 0  1  2  3  4  4  5  6  6  7  8  8  8  8  9  9 10 10 11 11 12 12 12 12 
 0  1  2  3  4  4  5  6  7  7  8  8  8  8  9  9 10 11 11 11 12 13 13 13 
 0  1  2  3  4  4  5  6  7  8  8  9  9  9  9  9 10 11 12 12 12 13 13 13 
 0  1  2  3  4  5  5  6  7  8  8  9  9 10 10 10 10 11 12 13 13 13 13 14 
 0  1  2  3  4  5  6  6  7  8  9  9  9 10 11 11 11 11 12 13 14 14 14 14 
 0  1  2  3  4  5  6  6  7  8  9  9  9 10 11 11 12 12 12 13 14 15 15 15 
 0  1  2  3  4  5  6  6  7  8  9  9  9 10 11 11 12 13 13 13 14 15 16 16 

Length of LCS: 16
LCS is: GCCCTAAGCTTAGCTT


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

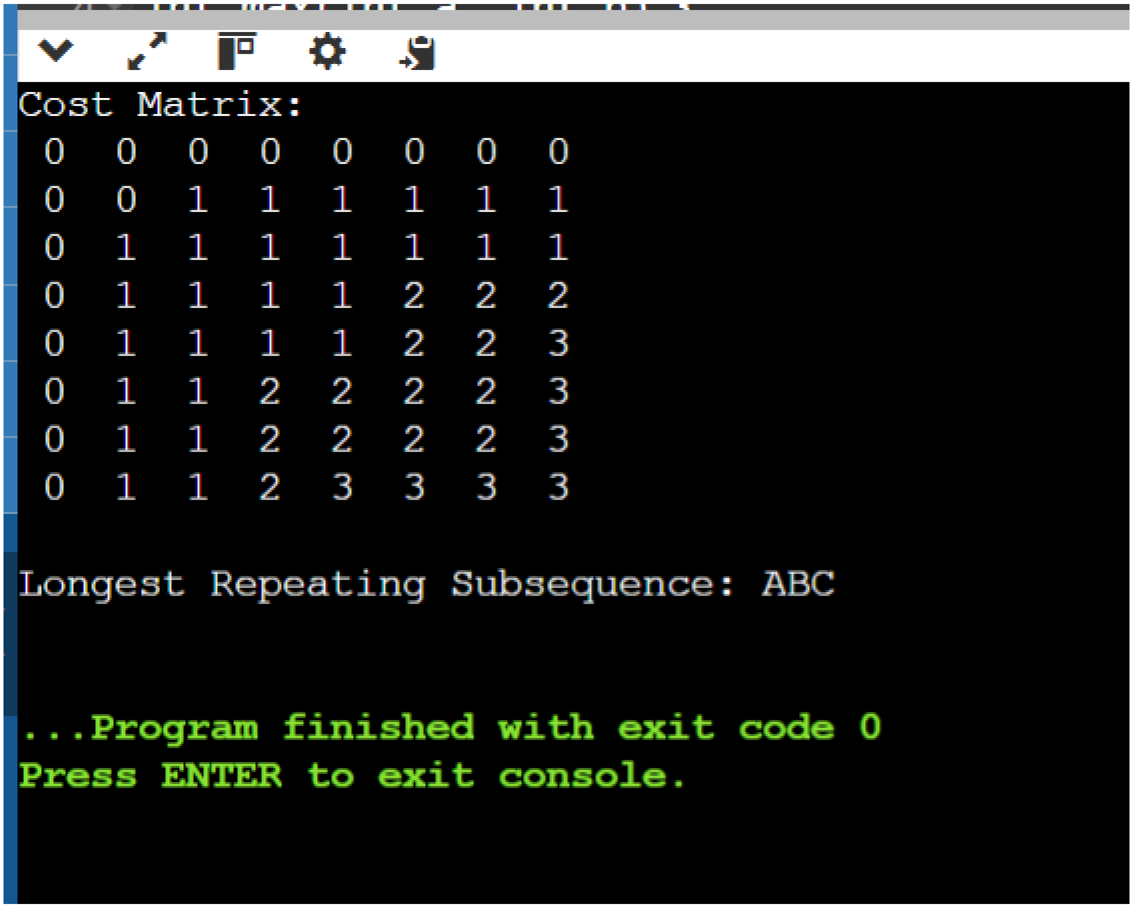
Task : 2

Code :



```c
#include <stdio.h>
#include <string.h>
int max(int a, int b) {
    return (a > b) ? a : b;
}
int main() {
    char S[] = "AABCBDC";
    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] = 1 + L[i - 1][j - 1];
            else
                L[i][j] = max(L[i - 1][j], L[i][j - 1]);
        }
    }
    printf("Cost Matrix:\n");
    for (int i = 0; i <= n; i++) {
        for (int j = 0; j <= n; j++)
            printf("%2d ", L[i][j]);
        printf("\n");
    }
    int index = L[n][n];
    char lrs[index + 1];
    lrs[index] = '\0';
    int i = n, j = n;
    while (i > 0 && j > 0) {
        if (L[i][j] == L[i - 1][j - 1] + 1 && S[i - 1] == S[j - 1] && i != j) {
            lrs[index - 1] = S[i - 1];
            i--; j--; index--;
        } else if (L[i - 1][j] > L[i][j - 1])
            i--;
        else
            j--;
    }
    printf("\nLongest Repeating Subsequence: %s\n", lrs);
    return 0;
}
```

Code :

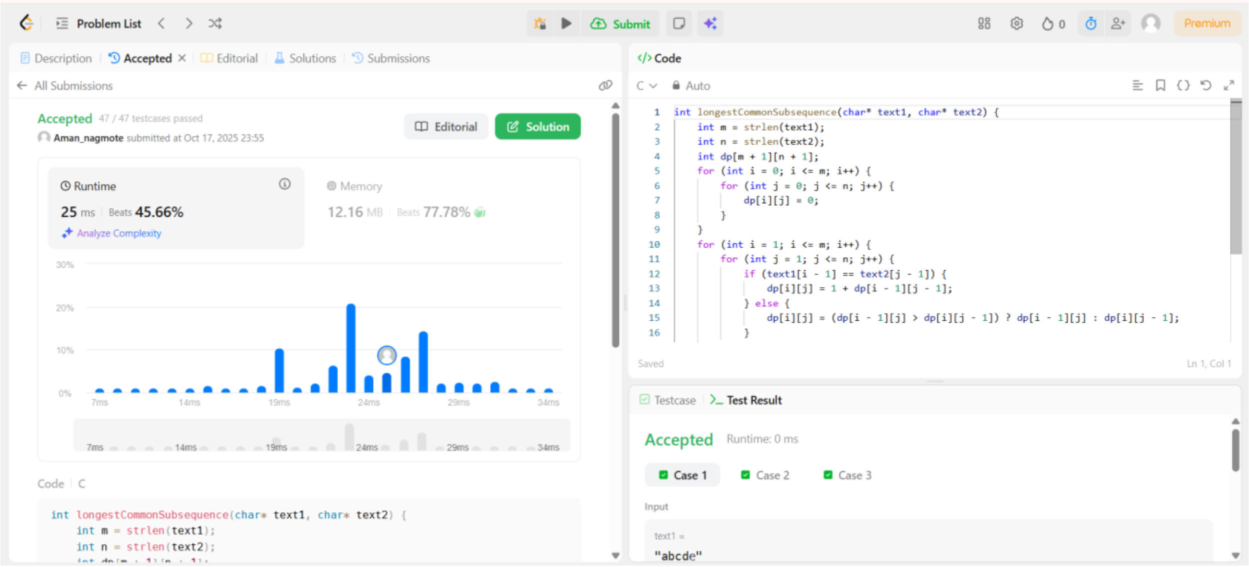


```
Cost Matrix:
 0   0   0   0   0   0   0   0
 0   0   1   1   1   1   1   1
 0   1   1   1   1   1   1   1
 0   1   1   1   1   2   2   2
 0   1   1   1   1   2   2   3
 0   1   1   2   2   2   2   3
 0   1   1   2   2   2   2   3
 0   1   1   2   3   3   3   3

Longest Repeating Subsequence: ABC


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

Leet code submission:



Accepted 47 / 47 testcases passed

Aman_nagmote submitted at Oct 17, 2025 23:55

Runtime

25 ms | Beats 45.66%

Memory

12.16 MB | Beats 77.78%

Code | C

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

```c
int longestCommonSubsequence(char* text1, char* text2) {
    int m = strlen(text1);
    int n = strlen(text2);
    int dp[m + 1][n + 1];
    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] = 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];
            }
```

Accepted Runtime: 0 ms

text1 =
"abcde"

My git hub link for pract 5 repo : https://github.com/24tiwaria2-code/DAA-