PRACTICAL NO. 5

Name: Isha Raut

Section: A4

Batch : B2

Roll No: 24

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

TASK 1:-

TASK 1: Find the similarity between the given X and Y sequence.

X=AGCCCTAAGGGCTACCTAGCTT

Y= GACAGCCTACAAGCGTTAGCTTG

Output: Cost matrix with all costs and direction, final cost of LCS and the LCS.

Length of LCS=16

CODE:-

```
X="AGCCCTAAGGGCTACCTAGCTT"
Y= "GACAGCCTACAAGCGTTAGCTTG"
m = len(X) n = len(Y)
c = [] b = [] for i
in range(m+1):
    c.append([0] * (n+1))
    b.append(['-'] * (n+1))
def print_matrices(): print("\nMatrix c (lengths)
and directions:") print("   ", end="") for ch in
Y: print(f" {ch} ", end="")
    print() for i in
    range(m+1): if i ==
    0: print(" ", end="
    ")
        else:
            print(f"{X[i-1]} ", end="")
        for j in range(n+1):
            if i == 0 or j == 0:
                print(f" {c[i][j]:2} ", end="")
            else: print(f"{c[i][j]}{b[i][j]} ",
                end="")
```

```
        print()

print("Initial matrices:")
print_matrices()
```

```
Initial matrices:

Matrix c (lengths) and directions:
      G  A  C  A  G  C  C  T  A  C  A  A  G  C  G  T  T  A  G  C  T  T  G
   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
A  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
G  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
C  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
C  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
C  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
T  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
A  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
A  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
G  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
G  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
G  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
C  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
T  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
A  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
C  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
C  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
T  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
A  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
G  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
C  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
T  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
T  0 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
```

```
for i in range(1, m+1):
    for j in range(1, n+1):
        if X[i-1] == Y[j-1]: c[i][j]
            = c[i-1][j-1] + 1 b[i][j]
            = '↖'
        elif c[i-1][j] >= c[i][j-1]:
            c[i][j] = c[i-1][j]
            b[i][j] = '↑'
        else:
            c[i][j]    =    c[i][j-1]
            b[i][j]       =       '←'
            print(f"\nAfter  updating
            c[{i}][{j}]        (X[{i-
            1}]='{X[i-1]}',
Y[{j-1}]='{Y[j-1]}'):")
        print_matrices()

print(f"\nLength of LCS = {c[m][n]}")
```

```
G   0  1↖ 2↑ 2↑ 3↑ 4↖ 4← 4↑ 5↑ 6↑ 6↑ 7↑ 7↑ 8↖ 8← 8↖ 8← 8← 8← 8↖ 8← 8← 8← 8↖
G   0  1↖ 2↑ 2↑ 3↑ 4↖ 4↑ 4↑ 5↑ 6↑ 6↑ 7↑ 7↑ 8↖ 8↑ 9↖ 9← 9← 9← 9↖ 9← 9← 9← 9↖
G   0  1↖ 2↑ 2↑ 3↑ 4↖ 4↑ 4↑ 5↑ 6↑ 6↑ 7↑ 7↑ 8↖ 8↑ 9↖ 9↑ 9↑ 9↑ 10↖ 10← 10← 10← 10↖
C   0  1↑ 2↑ 3↖ 3↑ 4↑ 5↖ 5↖ 5↑ 6↑ 7↖ 7↑ 7↑ 8↑ 9↖ 9↑ 9↑ 9↑ 9↑ 10↑ 11↖ 11← 11← 11←
T   0  1↑ 2↑ 3↑ 3↑ 4↑ 5↑ 5↑ 6↖ 6↑ 7↑ 7↑ 7↑ 8↑ 9↑ 9↑ 10↖ 10↖ 10← 10↑ 11↑ 12↖ 12↖ 12←
A   0  1↑ 2↖ 3↑ 4↖ 4↑ 5↑ 5↑ 6↑ 7↖ 7↑ 8↖ 8↖ 8↑ 9↑ 9↑ 10↑ 10↑ 11↖ 11← 11↑ 12↑ 12↑ 12↑
C   0  1↑ 2↑ 3↖ 4↑ 4↑ 5↖ 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
C   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
T   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
A   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
G   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
C   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
T   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
T   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
```

After updating c[15][7] (X[14]='C', Y[6]='C'):

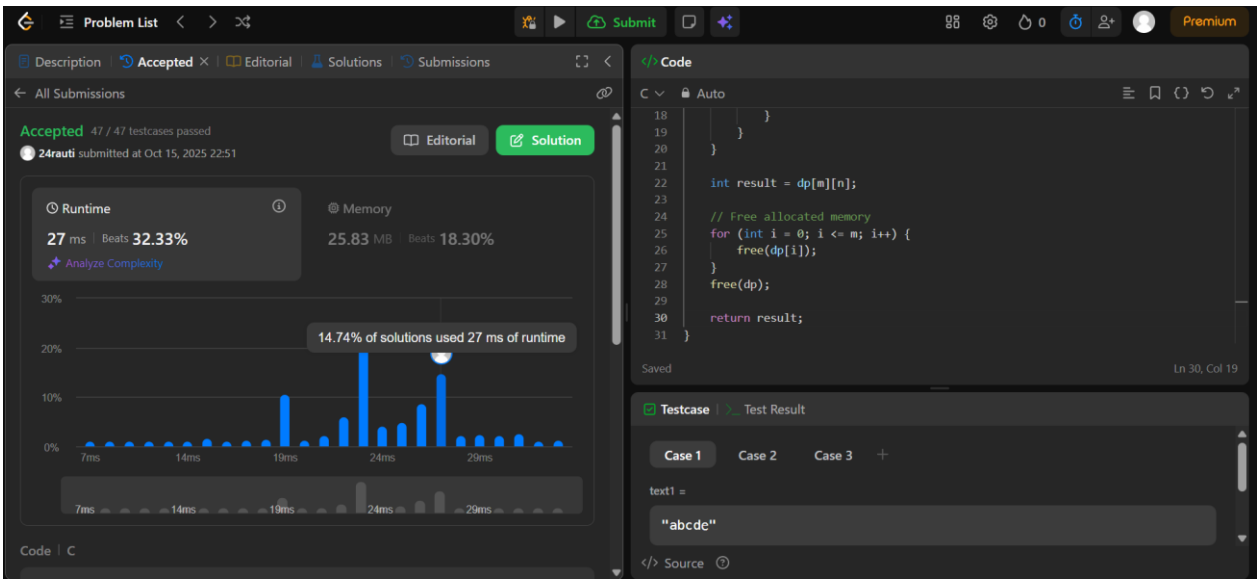
```
Matrix c (lengths) and directions:
      G   A   C   A   G   C   C   T   A   C   A   A   G   C   G   T   T   A   G   C   T   T   G
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
A   0  0↑ 1↖ 1← 1↖ 1← 1← 1← 1← 1↖ 1← 1↖ 1↖ 1← 1← 1← 1← 1← 1↖ 1← 1← 1← 1← 1←
G   0  1↖ 1↑ 1↑ 1↑ 2↖ 2← 2← 2← 2← 2← 2← 2← 2↖ 2← 2↖ 2← 2← 2← 2↖ 2← 2← 2← 2↖
C   0  1↑ 1↑ 2↖ 2← 2↑ 3↖ 3↖ 3← 3← 3↖ 3← 3← 3← 3↖ 3← 3← 3← 3← 3← 3↖ 3← 3← 3←
C   0  1↑ 1↑ 2↖ 2↑ 2↑ 3↖ 4↖ 4← 4← 4↖ 4← 4← 4← 4↖ 4← 4← 4← 4← 4← 4↖ 4← 4← 4←
C   0  1↑ 1↑ 2↖ 2↑ 2↑ 3↖ 4↖ 4↑ 4↑ 5↖ 5← 5← 5← 5↖ 5← 5← 5← 5← 5← 5↖ 5← 5← 5←
T   0  1↑ 1↑ 2↑ 2↑ 2↑ 3↑ 4↑ 5↖ 5← 5↑ 5↑ 5↑ 5↑ 5↑ 5↑ 6↖ 6↖ 6← 6← 6← 6↖ 6↖ 6←
A   0  1↑ 2↖ 2↑ 3↖ 3← 3↑ 4↑ 5↑ 6↖ 6← 6↖ 6↖ 6← 6← 6← 6↑ 6↑ 7↖ 7← 7← 7← 7← 7←
A   0  1↑ 2↖ 2↑ 3↖ 3↑ 3↑ 4↑ 5↑ 6↖ 6↑ 7↖ 7↖ 7← 7← 7← 7← 7← 7↖ 7↑ 7↑ 7↑ 7↑ 7↑
G   0  1↖ 2↑ 2↑ 3↑ 4↖ 4← 4↑ 5↑ 6↑ 6↑ 7↑ 7↑ 8↖ 8← 8↖ 8← 8← 8← 8↖ 8← 8← 8← 8↖
G   0  1↖ 2↑ 2↑ 3↑ 4↖ 4↑ 4↑ 5↑ 6↑ 6↑ 7↑ 7↑ 8↖ 8↑ 9↖ 9← 9← 9← 9↖ 9← 9← 9← 9↖
G   0  1↖ 2↑ 2↑ 3↑ 4↖ 4↑ 4↑ 5↑ 6↑ 6↑ 7↑ 7↑ 8↖ 8↑ 9↖ 9↑ 9↑ 9↑ 10↖ 10← 10← 10← 10↖
C   0  1↑ 2↑ 3↖ 3↑ 4↑ 5↖ 5↖ 5↑ 6↑ 7↖ 7↑ 7↑ 8↑ 9↖ 9↑ 9↑ 9↑ 9↑ 10↑ 11↖ 11← 11← 11←
T   0  1↑ 2↑ 3↑ 3↑ 4↑ 5↑ 5↑ 6↖ 6↑ 7↑ 7↑ 7↑ 8↑ 9↑ 9↑ 10↖ 10↖ 10← 10↑ 11↑ 12↖ 12↖ 12←
A   0  1↑ 2↖ 3↑ 4↖ 4↑ 5↑ 5↑ 6↑ 7↖ 7↑ 8↖ 8↖ 8↑ 9↑ 9↑ 10↑ 10↑ 11↖ 11← 11↑ 12↑ 12↑ 12↑
C   0  1↑ 2↑ 3↖ 4↑ 4↑ 5↖ 6↖ 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
C   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
T   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
A   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
G   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
C   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
T   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
T   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
```

After updating c[15][8] (X[14]='C', Y[7]='T'):

```
After updating c[15][8] (X[14]= C , Y[7]= T ):

Matrix c (lengths) and directions:
      G   A   C   A   G   C   C   T   A   C   A   A   G   C   G   T   T   A   G   C   T   T   G
      0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
A   0  0↑ 1↖ 1← 1↖ 1← 1← 1← 1← 1↖ 1← 1↖ 1↖ 1← 1← 1← 1← 1← 1↖ 1← 1← 1← 1← 1←
G   0  1↖ 1↑ 1↑ 1↑ 2↖ 2← 2← 2← 2← 2← 2← 2← 2↖ 2← 2↖ 2← 2← 2← 2↖ 2← 2← 2← 2↖
C   0  1↑ 1↑ 2↖ 2← 2↑ 3↖ 3↖ 3← 3← 3↖ 3← 3← 3↖ 3← 3← 3← 3← 3← 3↖ 3← 3← 3←
C   0  1↑ 1↑ 2↖ 2↑ 2↑ 3↖ 4↖ 4← 4← 4↖ 4← 4← 4← 4↖ 4← 4← 4← 4← 4← 4↖ 4← 4← 4←
C   0  1↑ 1↑ 2↖ 2↑ 2↑ 3↖ 4↖ 4↑ 4↑ 5↖ 5← 5← 5← 5↖ 5← 5← 5← 5← 5← 5↖ 5← 5← 5←
T   0  1↑ 1↑ 2↑ 2↑ 2↑ 3↑ 4↑ 5↖ 5← 5↑ 5↑ 5↑ 5↑ 5↑ 5↑ 6↖ 6↖ 6← 6← 6← 6↖ 6↖ 6←
A   0  1↑ 2↖ 2↑ 3↖ 3← 3↑ 4↑ 5↑ 6↖ 6← 6↖ 6↖ 6← 6← 6← 6↑ 6↑ 7↖ 7← 7← 7← 7← 7←
A   0  1↑ 2↖ 2↑ 3↖ 3↑ 3↑ 4↑ 5↑ 6↖ 6↑ 7↖ 7↖ 7← 7← 7← 7← 7← 7↖ 7↑ 7↑ 7↑ 7↑ 7↑
G   0  1↖ 2↑ 2↑ 3↑ 4↖ 4← 4↑ 5↑ 6↑ 6↑ 7↑ 7↑ 8↖ 8← 8↖ 8← 8← 8← 8↖ 8← 8← 8← 8↖
G   0  1↖ 2↑ 2↑ 3↑ 4↖ 4↑ 4↑ 5↑ 6↑ 6↑ 7↑ 7↑ 8↖ 8↑ 9↖ 9← 9← 9← 9↖ 9← 9← 9← 9↖
G   0  1↖ 2↑ 2↑ 3↑ 4↖ 4↑ 4↑ 5↑ 6↑ 6↑ 7↑ 7↑ 8↖ 8↑ 9↖ 9↑ 9↑ 9↑ 10↖ 10← 10← 10← 10↖
C   0  1↑ 2↑ 3↖ 3↑ 4↑ 5↖ 5↖ 5↑ 6↑ 7↖ 7↑ 7↑ 8↑ 9↖ 9↑ 9↑ 9↑ 9↑ 10↑ 11↖ 11← 11← 11←
T   0  1↑ 2↑ 3↑ 3↑ 4↑ 5↑ 5↑ 6↖ 6↑ 7↑ 7↑ 7↑ 8↑ 9↑ 9↑ 10↖ 10↖ 10← 10↑ 11↑ 12↖ 12↖ 12←
A   0  1↑ 2↖ 3↑ 4↖ 4↑ 5↑ 5↑ 6↑ 7↖ 7↑ 8↖ 8↖ 8↑ 9↑ 9↑ 10↑ 10↑ 11↖ 11← 11↑ 12↑ 12↑ 12↑
C   0  1↑ 2↑ 3↖ 4↑ 4↑ 5↖ 6↖ 6↑ 7↑ 8↖ 8↑ 8↑ 8↑ 9↑ 9↑ 10↑ 10↑ 11↑ 11↑ 12↑ 12↑ 13↖
C   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
T   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
A   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
G   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
C   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
T   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-
T   0  0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0- 0-

After updating c[22][23] (X[21]='T', Y[22]='G'):

Matrix c (lengths) and directions:
      G   A   C   A   G   C   C   T   A   C   A   A   G   C   G   T   T   A   G   C   T   T   G
      0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
A   0  0↑ 1↖ 1← 1↖ 1← 1← 1← 1← 1↖ 1← 1↖ 1↖ 1← 1← 1← 1← 1← 1↖ 1← 1← 1← 1← 1←
G   0  1↖ 1↑ 1↑ 1↑ 2↖ 2← 2← 2← 2← 2← 2← 2← 2↖ 2← 2↖ 2← 2← 2← 2↖ 2← 2← 2← 2↖
C   0  1↑ 1↑ 2↖ 2← 2↑ 3↖ 3↖ 3← 3← 3↖ 3← 3← 3↖ 3← 3← 3← 3← 3← 3↖ 3← 3← 3←
C   0  1↑ 1↑ 2↖ 2↑ 2↑ 3↖ 4↖ 4← 4← 4↖ 4← 4← 4← 4↖ 4← 4← 4← 4← 4← 4↖ 4← 4← 4←
C   0  1↑ 1↑ 2↖ 2↑ 2↑ 3↖ 4↖ 4↑ 4↑ 5↖ 5← 5← 5← 5↖ 5← 5← 5← 5← 5← 5↖ 5← 5← 5←
T   0  1↑ 1↑ 2↑ 2↑ 2↑ 3↑ 4↑ 5↖ 5← 5↑ 5↑ 5↑ 5↑ 5↑ 5↑ 6↖ 6↖ 6← 6← 6← 6↖ 6↖ 6←
A   0  1↑ 2↖ 2↑ 3↖ 3← 3↑ 4↑ 5↑ 6↖ 6← 6↖ 6↖ 6← 6← 6← 6↑ 6↑ 7↖ 7← 7← 7← 7← 7←
A   0  1↑ 2↖ 2↑ 3↖ 3↑ 3↑ 4↑ 5↑ 6↖ 6↑ 7↖ 7↖ 7← 7← 7← 7← 7↖ 7↑ 7↑ 7↑ 7↑ 7↑
G   0  1↖ 2↑ 2↑ 3↑ 4↖ 4← 4↑ 5↑ 6↑ 6↑ 7↑ 7↑ 8↖ 8← 8↖ 8← 8← 8← 8↖ 8← 8← 8← 8↖
G   0  1↖ 2↑ 2↑ 3↑ 4↖ 4↑ 4↑ 5↑ 6↑ 6↑ 7↑ 7↑ 8↖ 8↑ 9↖ 9← 9← 9← 9↖ 9← 9← 9← 9↖
G   0  1↖ 2↑ 2↑ 3↑ 4↖ 4↑ 4↑ 5↑ 6↑ 6↑ 7↑ 7↑ 8↖ 8↑ 9↖ 9↑ 9↑ 9↑ 10↖ 10← 10← 10← 10↖
C   0  1↑ 2↑ 3↖ 3↑ 4↑ 5↖ 5↖ 5↑ 6↑ 7↖ 7↑ 7↑ 8↑ 9↖ 9↑ 9↑ 9↑ 9↑ 10↑ 11↖ 11← 11← 11←
T   0  1↑ 2↑ 3↑ 3↑ 4↑ 5↑ 5↑ 6↖ 6↑ 7↑ 7↑ 7↑ 8↑ 9↑ 9↑ 10↖ 10↖ 10← 10↑ 11↑ 12↖ 12↖ 12←
A   0  1↑ 2↖ 3↑ 4↖ 4↑ 5↑ 5↑ 6↑ 7↖ 7↑ 8↖ 8↖ 8↑ 9↑ 9↑ 10↑ 10↑ 11↖ 11← 11↑ 12↑ 12↑ 12↑
C   0  1↑ 2↑ 3↖ 4↑ 4↑ 5↖ 6↖ 6↑ 7↑ 8↖ 8↑ 8↑ 8↑ 9↖ 9↑ 10↑ 10↑ 11↑ 11↑ 12↖ 12↑ 12↑ 12↑
C   0  1↑ 2↑ 3↖ 4↑ 4↑ 5↖ 6↖ 6↑ 7↑ 8↖ 8↑ 8↑ 8↑ 9↖ 9↑ 10↑ 10↑ 11↑ 11↑ 12↖ 12↑ 12↑ 12↑
T   0  1↑ 2↑ 3↑ 4↑ 4↑ 5↑ 6↑ 7↖ 7↑ 8↑ 8↑ 8↑ 8↑ 9↑ 9↑ 10↖ 11↖ 11↑ 11↑ 12↑ 13↖ 13↖ 13←
A   0  1↑ 2↖ 3↑ 4↖ 4↑ 5↑ 6↑ 7↑ 8↖ 8↑ 9↖ 9↖ 9← 9↑ 9↑ 10↑ 11↑ 12↖ 12← 12↑ 13↑ 13↑ 13↑
G   0  1↖ 2↑ 3↑ 4↑ 5↖ 5↑ 6↑ 7↑ 8↑ 8↑ 9↑ 9↑ 10↖ 10← 10↖ 10↑ 11↑ 12↑ 13↖ 13← 13↑ 13↑ 14↖
C   0  1↑ 2↑ 3↖ 4↑ 5↑ 6↖ 6↖ 7↑ 8↑ 9↖ 9↑ 9↑ 10↑ 11↖ 11← 11← 11↑ 12↑ 13↑ 14↖ 14← 14← 14↑
T   0  1↑ 2↑ 3↑ 4↑ 5↑ 6↑ 6↑ 7↖ 8↑ 9↑ 9↑ 9↑ 10↑ 11↑ 11↑ 12↖ 12↖ 12↑ 13↑ 14↑ 15↖ 15↖ 15←
T   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
```

LeetCode Assesment:

https://leetcode.com/problems/longest-common-subsequence/submissions/1802644206/



TASK-2:

TASK-2: Find the longest repeating subsequence (LRS). Consider it as a variation of the longest common subsequence (LCS) problem.
Let the given string be S. You need to find the LRS within S. To use the LCS framework, you effectively compare S with itself. So, consider string1 = S and string2 = S. Example:
AABCBDC
LRS= ABC or ABD

CODE:-

```
def LRS(a, b):
n = len(a)      m
= len(b)


    c = [[0 for _ in range(m + 1)] for _ in range(n + 1)]


    for i in range(1, n + 1):
for j in range(1, m + 1):
            if a[i - 1] == b[j - 1] and i != j:
c[i][j] = 1 + c[i - 1][j - 1]              else:
                c[i][j] = max(c[i - 1][j], c[i][j - 1])
```

```
    i, j = n, m
lrs_subsequence = []
     while i > 0 and j >
0:
        if a[i - 1] == b[j - 1] and i != j:
lrs_subsequence.append(a[i - 1])
i -= 1              j -= 1
        elif c[i - 1][j] >= c[i][j - 1]:
i -= 1          else:
j -= 1

lrs_subsequence.reverse()
     return ''.join(lrs_subsequence),
c[n][m]


a = "AABCBDC" b = a
lrs_subsequence, length = LRS(a, b)
 print(f"Longest Repeated Subsequence:
{lrs_subsequence}") print(f"Length of LRS: {length}")
```

OUTPUT:-

```
Longest Repeated Subsequence: ABC
Length of LRS: 3
```