Sub: Algorithm Analysis and Design

A subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements. Longest common subsequence (LCS) of 2 sequences is a subsequence, with maximal length, which is common to both the sequences.

Given two sequence of integers, P = <M, N, O, M> and Q = <M, L, N, O, M>, find any one longest common subsequence.

In case multiple solutions exist, print any of them. It is guaranteed that at least one non-empty common subsequence will exist.

CODE:

```
def con_int(strings):
    v=''
    w=0
    try:
        int(strings)
        return strings
    except:
        for i in strings:
            try:
                w=int(i)
                v+=str(w)
            except:
                continue
        return int(v)
```

```
def display_LCS(d, a, i, j):

    if i==0 or j==0:
        return
    elif d[i][j]=='↖':
        display_LCS(d, a, i-1, j-1)
        print(a[i])
    elif d[i][j]=='↑':
        display_LCS(d, a, i-1, j)
    else:
        display_LCS(d, a, i, j-1)



def find_LCS(a, b):
    aa=0
    A=[0]
    B=[0]

    for i in a:
        A.append(i)

    for j in b:
        B.append(j)

    matrix=[]
    arrow=[]
```

```
m=len(A)
n=len(B)



for i in range(m):
    matr=[]
    for j in range(n):
        matr.append(0)
    matrix.append(matr)

for i in range(m):
    mat=[]
    for j in range(n):
        mat.append(0)
    arrow.append(mat)

for i in range(m):
    for j in range(n):
        if i == 0 or j == 0:
            matrix[i][j]=0
        else:
            if A[i] == B[j]:
                matrix[i][j]=matrix[i-1][j-1]+1
            else:
                aa = max(matrix[i][j-1], matrix[i-1][j])
                matrix[i][j]=aa

for i in range(m):
```

```
            for j in range(n):
                if i == 0 or j == 0:
                    matrix[i][j]=0
                else:
                    if A[i] == B[j]:
                        matrix[i][j]=str(matrix[i][j])
                        matrix[i][j]+='↖'
                        arrow[i][j]='↖'
                    elif con_int(matrix[i-1][j]) >= con_int(matrix[i][j-1]):
                        matrix[i][j]=str(matrix[i][j])
                        matrix[i][j]+='↑'
                        arrow[i][j]='↑'
                    else:
                        matrix[i][j]=str(matrix[i][j])
                        matrix[i][j]+='←'
                        arrow[i][j]='←'

        return matrix, arrow, A



ll1=[str(i) for i in input('Enter first string: ').split(' ')]
ll2=[str(i) for i in input('Enter second string: ').split(' ')]
ii=len(ll1)
jj=len(ll2)

matri, arrows, ll3 = find_LCS(ll1, ll2)

print('\n')
```

```
print('final Matrix: ')
for i in matri:
    print(*i)

print('\n')
print('Arrow Matrix:')
for j in arrows:
    print(*j)

print('\n')
print('LCS: ')
display_LCS(arrows, ll3, ii, jj)
```

OUTPUT:

Tejas Tripathi

```
In [81]: runfile('C:/Users/Admin/study material/sem5/Practicals/Algorithms/Practical-11/LCS_algo.py',
wdir='C:/Users/Admin/study material/sem5/Practicals/Algorithms/Practical-11')

Enter first string: M N O M

Enter second string: M L N O M


final Matrix:
0 0 0 0 0 0
0 1↖ 1← 1← 1← 1↖
0 1↑ 1↑ 2↖ 2← 2←
0 1↑ 1↑ 2↑ 3↖ 3←
0 1↖ 1↑ 2↑ 3↑ 4↖


Arrow Matrix:
0 0 0 0 0 0
0 ↖ ← ← ← ↖
0 ↑ ↑ ↖ ← ←
0 ↑ ↑ ↑ ↖ ←
0 ↖ ↑ ↑ ↑ ↖


LCS:
M
N
O
M

In [82]:
```