**Program :**

#include <bits/stdc++.h>

using namespace std;

int L[10][10];

void display(string X, string Y, int m, int n){

    cout << "    ";

    for (int i = 0; i < m; i++)

        cout << X[i] << " ";

    cout << endl;

    for (int i = 0; i <= n; i++)

    {

        if (i > 0)

            cout << Y[i - 1] << " ";

        else

            cout << "  ";

        for (int j = 0; j <= m; j++)cout << L[j][i] << " ";

        cout << endl;

    }

}

int lcs(string X, string Y, int m, int n){

    int i, j;

    vector<char> lcs;

    for (i = 0; i <= m; i++){

        for (j = 0; j <= n; j++){

            if (i == 0 || j == 0) //initializing top left edgr with 0

                L[i][j] = 0;

            else if (X[i - 1] == Y[j - 1])//if equal add 1 to diag elem

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

            else                          // if unequal take max of upper or

                L[i][j] = max(L[i - 1][j], L[i][j - 1]);

        }

    }

    display(X,Y,m,n);

    i = m;

    j = n;

    cout << "LCS : ";

    while (i > 0 && j > 0)

    {

        if (L[i][j] == L[i][j - 1])

        {

            j = j - 1;

        }

        else if (L[i][j] == L[i - 1][j])

        {

            i = i - 1;

        }

        else if (L[i][j] == L[i - 1][j - 1] + 1)

        {

            lcs.push\_back(X[i - 1]);

            i = i - 1;

            j = j - 1;

        }

    }

    for (auto x = lcs.rbegin(); x != lcs.rend(); ++x)

        cout << \*x;

    return L[m][n];

}

int main()

{

    string str1, str2;

    cout << "Enter string 1 :";

    cin >> str1;

    cout << "Enter string 2 :";

    cin >> str2;

    int m = str1.length();

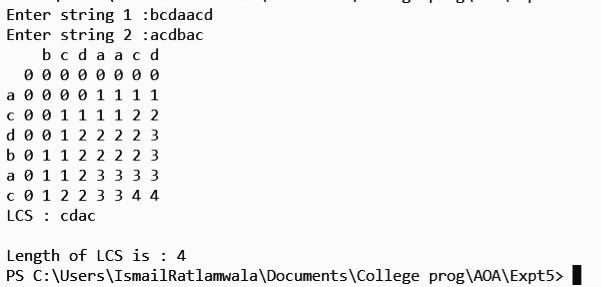
    int n = str2.length();

    printf("\n\nLength of LCS is : %d\n", lcs(str1, str2, m, n));

    return 0;

}

**Output :**

****

**Program :**

#include <bits/stdc++.h>

using namespace std;

struct Edge {

    int src, dest, weight;

};

struct Graph {

    int V, E;

    struct Edge\* edge;

};

struct Graph\* createGraph(int V, int E)

{

    struct Graph\* graph = new Graph;

    graph->V = V;

    graph->E = E;

    graph->edge = new Edge[E];

    return graph;

}

void printArr(int dist[], int n)

{

    printf("Vertex Distance from Source\n");

    for (int i = 0; i < n; ++i)

        printf("%d \t\t %d\n", i, dist[i]);

}

void BellmanFord(struct Graph\* graph, int src)

{

    int V = graph->V;

    int E = graph->E;

    int dist[V];

    for (int i = 0; i < V; i++)

        dist[i] = INT\_MAX;

    dist[src] = 0;

    for (int i = 1; i <= V - 1; i++) {

        for (int j = 0; j < E; j++) {

            int u = graph->edge[j].src;

            int v = graph->edge[j].dest;

            int weight = graph->edge[j].weight;

            if (dist[u] != INT\_MAX

                && dist[u] + weight < dist[v])

                dist[v] = dist[u] + weight;

        }

    }

    for (int i = 0; i < E; i++) {

        int u = graph->edge[i].src;

        int v = graph->edge[i].dest;

        int weight = graph->edge[i].weight;

        if (dist[u] != INT\_MAX

            && dist[u] + weight < dist[v]) {

            printf("Graph contains negative weight cycle");

            return;

        }

    }

    printArr(dist, V);

}

int main()

{

    int V,E;

    cout<<"Enter number of Vertices in the graph: ";

    cin>>V;

    cout<<"Enter number of Edges in the graph: ";

    cin>>E;

    struct Graph\* graph = createGraph(V, E);

    cout<<"Enter source, destination and weight :\n";

    for(int i=0;i<E;i++){

        cin>>graph->edge[i].src;

        cin>>graph->edge[i].dest;

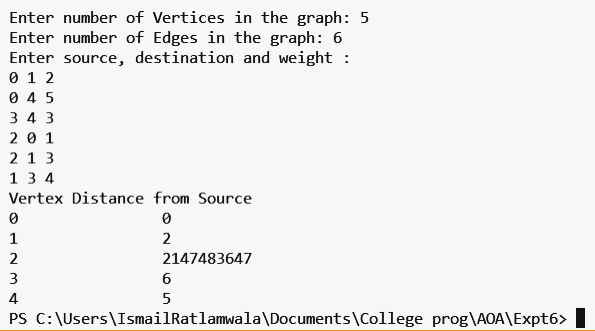
        cin>>graph->edge[i].weight;

    }

    BellmanFord(graph, 0);

}

**Output :**



**Program :**

#include <bits/stdc++.h>

using namespace std;

#define d 256

void search(string pat, string txt, int q) {

    int M = pat.length();

    int N = txt.length();

    int i, j;

    int p = 0,t=0,h=1;

    for (i = 0; i < M - 1; i++)

        h = (h \* d) % q;

    for (i = 0; i < M; i++){

        p = (d \* p + pat[i]) % q;

        t = (d \* t + txt[i]) % q;

    }

    for (i = 0; i <= N - M; i++){

        if ( p == t ){

            bool flag = true;

            for (j = 0; j < M; j++)

            {

                if (txt[i+j] != pat[j])

                {

                  flag = false;

                  break;

                }

            }

            if (j == M)

                cout<<"Pattern found at index "<< i<<endl;

        }

        if ( i < N-M ) {

            t = (d\*(t - txt[i]\*h) + txt[i+M])%q;

            if (t < 0)

            t = (t + q);

        }

    }

}

int main(){

    string txt,pat;

    cout<<"Enter text to be searched : ";

    cin>>txt;

    cout<<"Enter the pattern : ";

    cin>>pat;

    search(pat, txt, 101);

    return 0;

}

**Output :**

