**CSE2012-LAB**

NAME : O G RAGAVI

REG NO: 20BCE1988

EX NO: 6

**TITLE:** LONGEST COMMON SUBSEQUENCE

**LAB PRACTICE SHEET:**

**1.BRUTE FORCE APPROACH FOR FINDING LCS:**

**CODE:**

/\* A Naive recursive implementation of LCS problem \*/

#include <bits/stdc++.h>

using namespace std;

int lcs( char \*X, char \*Y, int m, int n )

{

if (m == 0 || n == 0)

return 0;

if (X[m-1] == Y[n-1])

return 1 + lcs(X, Y, m-1, n-1);

else

return max(lcs(X, Y, m, n-1), lcs(X, Y, m-1, n));

}

int main()

{

char X[] = "ARABIAN";

char Y[] = "INDIAN";

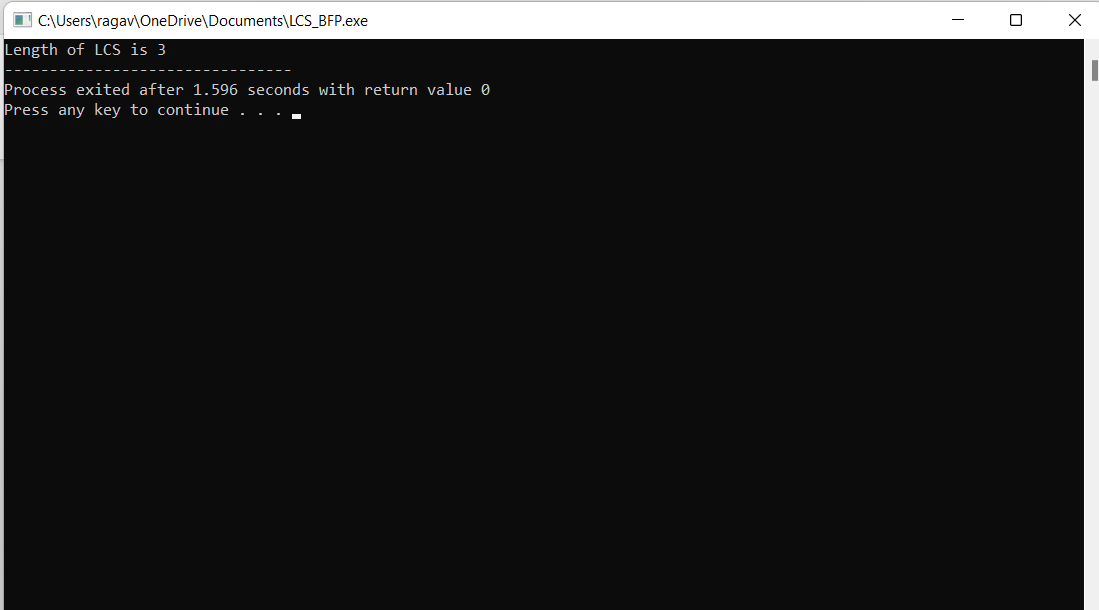
int m = strlen(X);

int n = strlen(Y);

cout<<"Length of LCS is "<< lcs( X, Y, m, n ) ;

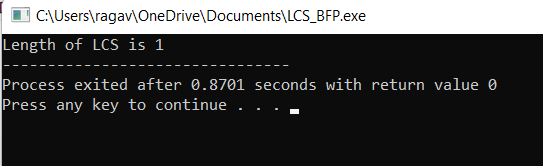
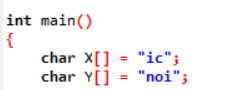
return 0;

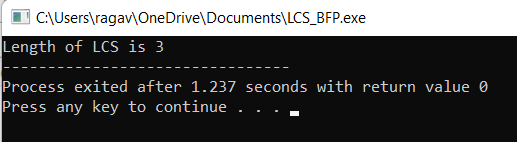
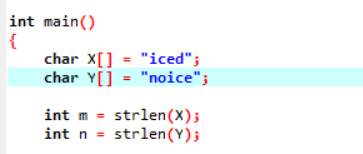
}

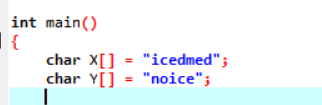
**OUTPUT:**

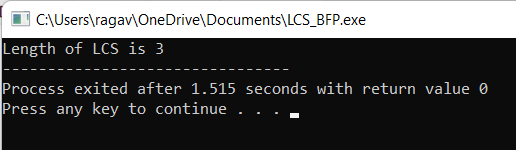
**2.RUNNING TIME COMPUTATION:**

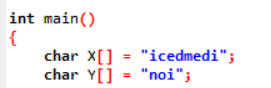
|  |  |  |
| --- | --- | --- |
| m | n | T1(p)(in seconds) |
| 2 | 3 | 0.8701 |
| 4 | 5 | 1.237 |
| 7 | 5 | 1.515 |
| 8 | 3 | 1.751 |
| 5 | 5 | 1.865 |

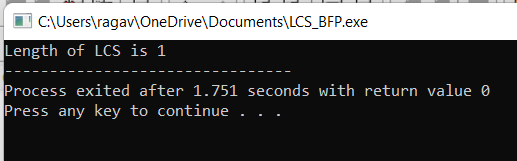


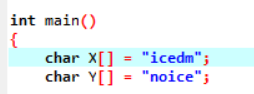


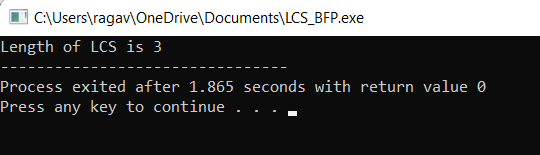












**3.DYNAMIC PROGRAMMING ALGORTIHM FOR LCS:**

CODE:

// Recursive routine

#include<iostream>

using namespace std;

#include<limits.h>

#include<vector>

int lcs\_length(string x, string y, int x\_index,int y\_index)

{

int m,n,i,j,l1,l2;

m = x.length();

n = y.length();

if((x\_index>=m)||(y\_index>=n))

return 0;

if(x[x\_index]==y[y\_index])

{

return lcs\_length(x,y,x\_index+1,y\_index+1)+1;

}

else

{

l1 = lcs\_length(x,y,x\_index,y\_index+1);

l2 = lcs\_length(x,y,x\_index+1,y\_index);

return l1>l2? l1: l2;

}

}

int main()

{

string x,y;

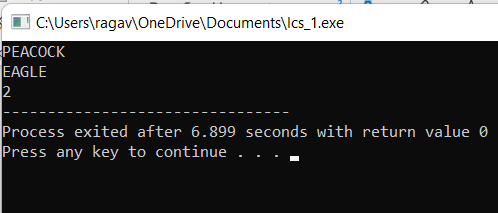
int i,j;

cin>>x>>y;

cout<<lcs\_length(x,y,0,0);

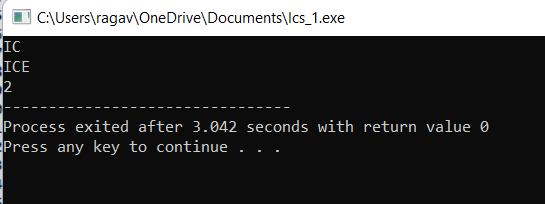
}

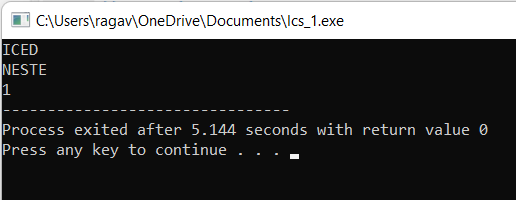
**OUTPUT:**

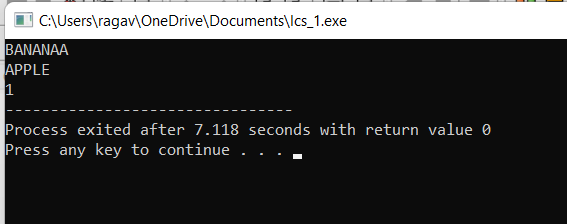


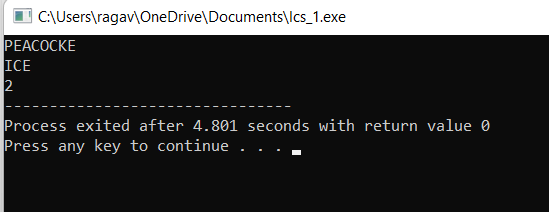
4.RUNNING TIME COMPUTATION:

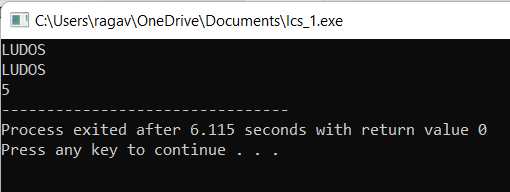
|  |  |  |
| --- | --- | --- |
| m | n | T2(p)(in seconds) |
| 2 | 3 | 3.042 |
| 4 | 5 | 5.144 |
| 7 | 5 | 7.188 |
| 8 | 3 | 4.801 |
| 5 | 5 | 6.115 |







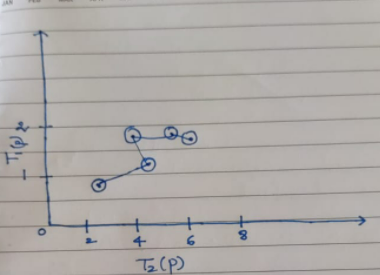




5.COMPARISON OF RUNNING TIME:

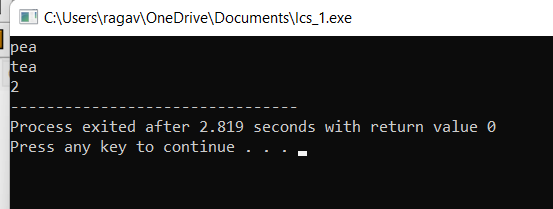
|  |  |  |  |
| --- | --- | --- | --- |
| m | n | T2(p)(in seconds) | T1(P) |
| 2 | 3 | 3.042 | 0.8701 |
| 4 | 5 | 5.144 | 1.237 |
| 7 | 5 | 7.188 | 1.515 |
| 8 | 3 | 4.801 | 1.751 |
| 5 | 5 | 6.115 | 1.865 |

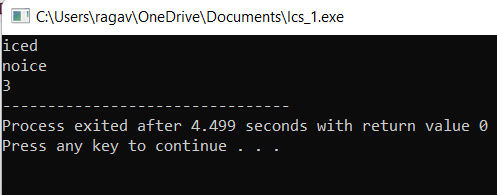
**6.GRAPH:**

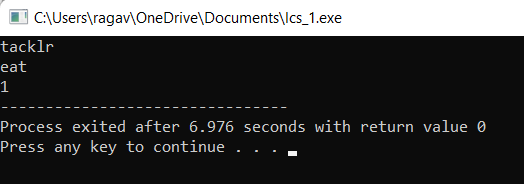
****

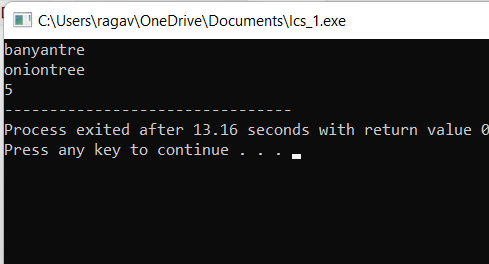
**7.RUNNING TIME COMPUTATION:**

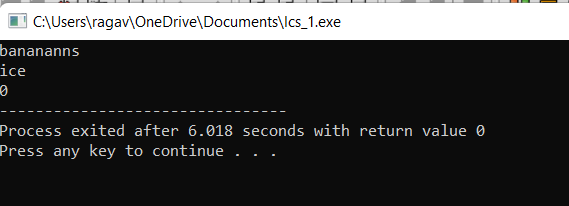
|  |  |  |  |
| --- | --- | --- | --- |
| m | n | T2(p) | O(mn) |
| 3 | 3 | 2.819 | 9 |
| 4 | 5 | 4.499 | 20 |
| 6 | 7 | 6.967 | 42 |
| 9 | 8 | 13.16 | 72 |
| 8 | 3 | 6.018 | 24 |











8.

|  |  |  |
| --- | --- | --- |
| m | n | O(mn) |
| 3 | 3 | 9 |
| 4 | 5 | 20 |
| 6 | 7 | 42 |
| 9 | 8 | 72 |
| 8 | 3 | 24 |

9.The longest common subsequence of ‘apple’ and ‘pebble’ is ‘ple’ and even the LCS of ‘apple’ and ‘nopelle’ is ‘ple’.So, the LCS of 2 pairs need not be unique.