```cpp
//Longest Common Subsequence
#include<iostream>
#include<fstream>
#include<stdlib.h>
#include<time.h>
#include<iomanip>
#include<bits/stdc++.h>

using namespace std;

long int cnt=0;

//LCS - Dynamic Programming
int LCS_Length_DP(char *x, char *y, int m, int n,char **b,
int **c)
{
    //m and n are the Length of x and y resp.

    for(int i=1;i<=m;i++)
    {
        c[i][0] = 0;
        cnt++;
    }
    for(int j=0;j<=n;j++)
    {
        c[0][j] = 0;
        cnt++;
    }

    for(int i=1;i<=m;i++)
    {
        for(int j=1;j<=n;j++)
        {
            cnt++;
            if(x[i]==y[j])
            {
                c[i][j] = c[i-1][j-1]+1;
                b[i][j] = 'C';
            }
            else if(c[i-1][j]>=c[i][j-1])
            {
                c[i][j] = c[i-1][j];
                b[i][j] = 'U';
            }
            else
            {
                c[i][j] = c[i][j-1];
                b[i][j] = 'L';
            }
        }
    }

    return c[m][n];

}

//Print LCS
void PrintLCS(char **b,char *x, int i, int j)
{
    if(i==0 || j==0)
    {
        return;
    }

    if(b[i][j]=='C')
    {
        PrintLCS(b,x,i-1,j-1);
        cout<<x[i];
    }
    else if(b[i][j]=='U')
    {
        PrintLCS(b,x,i-1,j);
    }
    else
    {
        PrintLCS(b,x,i,j-1);
    }

}


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

```cpp
}

//LCS - Divide & Conquer
int LCS_Length_DC( char *X, char *Y, int m, int n )
{
    cnt++;
    if (m == 0 || n == 0)
        return 0;
    if (X[m-1] == Y[n-1])
        return 1 + LCS_Length_DC(X, Y, m-1, n-1);
    else
        return max(LCS_Length_DC(X, Y, m, n-1),
LCS_Length_DC(X, Y, m-1, n));
}




int main()
{
    cout<<showpoint<<setprecision(12);

    int n,m;
    char *x;
    char *y;

    cout<<"\nEnter Length of First String: ";
    cin>>m;
    cout<<"\nEnter Length of Second String: ";
    cin>>n;



    x = new char[m+1];
    y = new char[n+1];


    ofstream outf;
    ifstream inf;

    srand((long int)clock());

    //Loading numbers to input file
    char t;
    outf.open("in1.txt");
    for(int i=1;i<=m;i++)
    {
//      while(((t=(rand()%255)+1)<65 || (t>90&&t<97) || t>122)); //For any alphabets
//      while(((t=(rand()%255)+1)<65 || (t>68&&t<97) || t>100)); //For only a,b,c,d & A,B,C,D
        while(((t=(rand()%255)+1)<'A') || (t>'A'&&t<'C') || (t>'C'&&t<'G') || (t>'G'&&t<'T') ||t>'T'); //For DNA Sequence
        outf<<"\t"<<t;
    }
    outf.close();
    outf.open("in2.txt");
    for(int i=1;i<=n;i++)
    {
//      while(((t=(rand()%255)+1)<65 || (t>90&&t<97) || t>122)); //For any alphabets
//      while(((t=(rand()%255)+1)<65 || (t>68&&t<97) || t>100)); //For only a,b,c,d & A,B,C,D
        while(((t=(rand()%255)+1)<'A') || (t>'A'&&t<'C') || (t>'C'&&t<'G') || (t>'G'&&t<'T') ||t>'T'); //For DNA Sequence
        outf<<"\t"<<t;
    }
    outf.close();

    //Reading input in array from input file

    inf.open("in1.txt");
    for(int i=1;i<=m;i++)
    {
        inf>>x[i];
    }
    inf.close();
    x[m+1] = '\0';
    x[0]=' ';
    inf.open("in2.txt");
    for(int i=1;i<=n;i++)
    {
        inf>>y[i];
    }
    inf.close();
    y[n+1] = '\0';
    y[0]=' ';

    cout<<"\n\nX: "<<x;
    cout<<"\n\nY: "<<y;


    char **b; //U - Up, L - Left & C - Cross
    int **c;

    b = new char*[m+1];
    for(int i=0;i<=m;i++)
        b[i] = new char[n+1];

    c = new int*[m+1];
    for(int i=0;i<=m;i++)
        c[i] = new int[n+1];

    int lcs_length=0;

    //LCS - Divide & Conquer
    cnt = 0;
    lcs_length = LCS_Length_DC(x,y,m,n);

    cout<<"\n\nLongest Common Sub Sequence Length (D & C): 
"<<lcs_length;
    cout<<"\nNumber of Active Operations: "<<cnt;

    //LCS - Dynamic Programming

    cnt=0;
    lcs_length = LCS_Length_DP(x,y,m,n,b,c);

    cout<<"\n\nLongest Common Sub Sequence Length (DP): 
"<<lcs_length;
    cout<<"\nNumber of Active Operations: "<<cnt;
    cout<<"\n\nLCS: ";
    PrintLCS(b,x,m,n);


    delete(b);
    delete(c);

}

```