UCS415 – Design and Analysis of AlgorithmsLab

Assignment 3

Name: Akshita Pathak
Roll No: 102203796
Subgroup: 2CO18

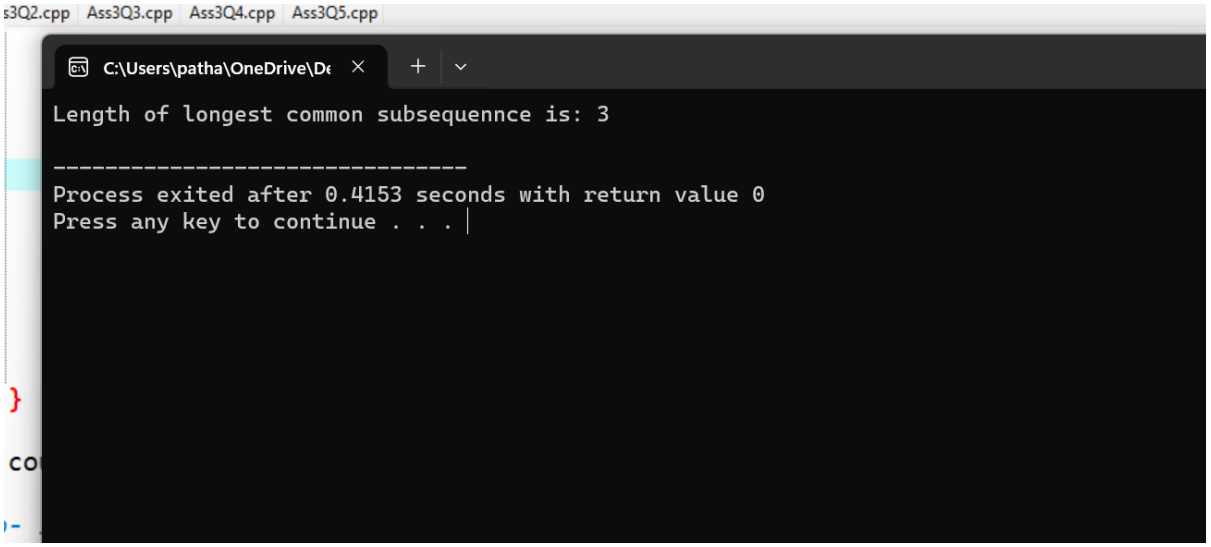
Write a program to implement the following using dynamic programming approach:

- **Longest Common Subsequence**

```
/*
1. Longest Common Subsequence
*/

#include<iostream>
using namespace std;



int main()
{
    string A="stone";
    string B="longest";

    int m=A.length();
    int n=B.length();

    int lcs[m+1][n+1];

    for(int i=0;i<=m;i++){
    for(int j=0;j<=n;j++){
        if(i==0 || j==0 ){
            lcs[i][j]=0;
        }
        else if(A[i-1]==B[j-1]){
            lcs[i][j]=1+lcs[i-1][j-1];
        }
        else{
            lcs[i][j]=max(lcs[i-1][j],lcs[i][j-1]);
        }
    }
    }
```

```
    cout<<"Length of longest common subsequennce is:
    "<<lcs[m][n]<<endl;

//op- 3 -- one

    return 0;
    }
```

OUTPUT:

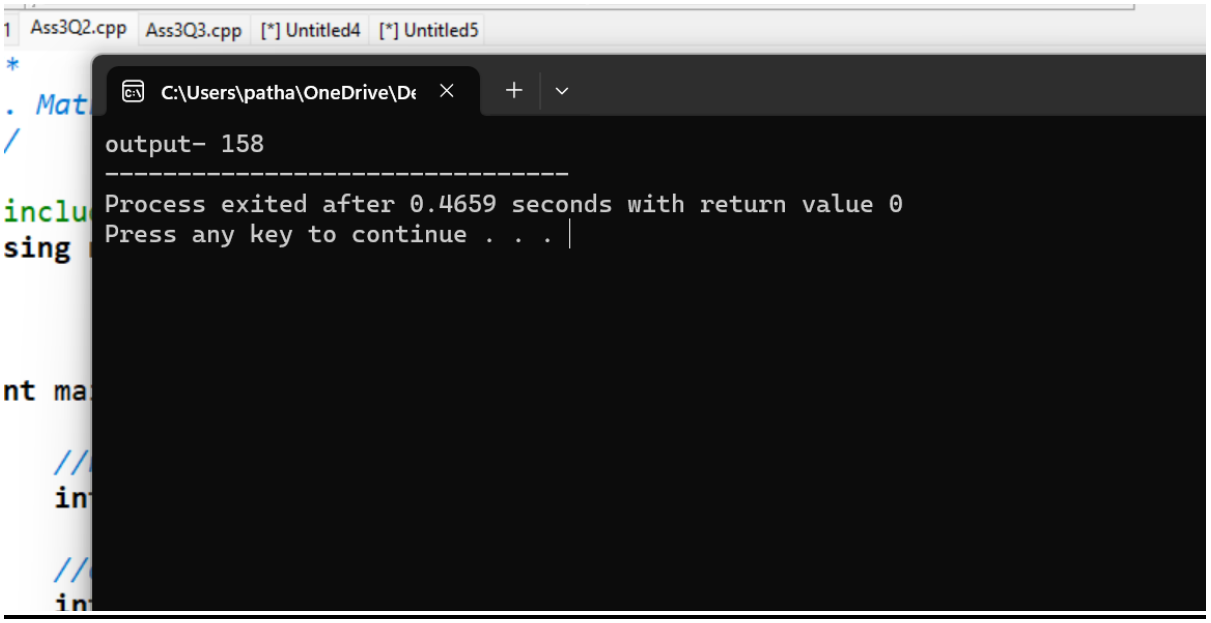


- **Matrix Chain Multiplication**

```
/*
2. Matrix Chain Multiplication
*/

#include<iostream>
using namespace std;




int main()
{
    //no of matrices+1
    int n=5;

    //array to store dimensions of matrices
    int d[]={5,4,6,2,7};

    //take 2d arrays to store result
    int c[5][5]={0};
    int k[5][5]={0};

    //take diff of j-1
    for(int diff=1;diff<n-1;diff++){
    //row
    for(int i=1; i<n-diff;i++){
        //clm
        int j=i+diff;
        int min=32767;
        int cost;

        //find minm
        for(int K=i;K<j;K++){
            cost=c[i][K]+c[K+1][j]+d[i-1]*d[K]*d[j];
            if(cost<min){
                min=cost;
                k[i][j]=K;
            }
```

```
        }
        c[i][j]=min;
    }
    }

    cout<<"output- "<<c[1][n-1];

    return 0;
}

    //op- 158
```

OUTPUT:

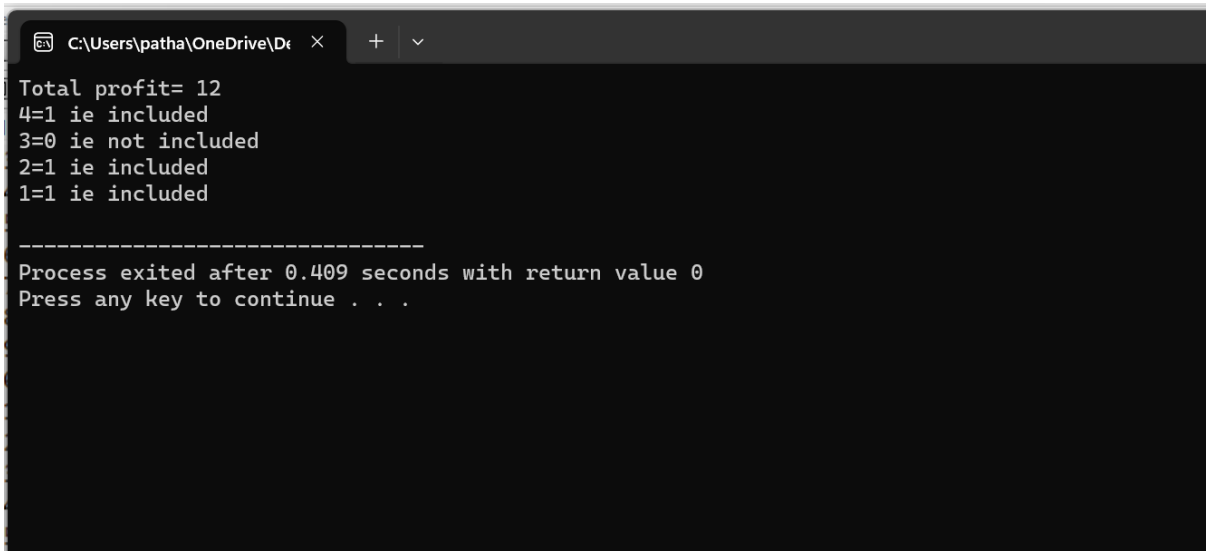


- **0/1 Knapsack Problem**

```
/*
3. 0/1 Knapsack Problem
*/

#include<iostream>
using namespace std;

int main()
{
    //array of profits
    int p[5]={0,2,4,7,10};
    //0 index just added

    //array of weights of objects
    int wt[5]={0,1,3,5,7};

    //knapsack capacity M
    int M=8;

    //no of objects
    int n=4;

    //2d array table where row-objects(n+1) and clm-capacity(M+1)
    int k[5][9];

    //i row
    for(int i=0;i<=n;i++){
    //w clm
    for(int w=0;w<=M;w++){
        //for all 0 indices- profit is 0
        if(i==0 || w==0){
            k[i][w]=0;
        }
        //if weight of object is less than capacity
        else if(wt[i]<=w){
            k[i][w]=max(k[i-1][w],k[i-1][w-wt[i]]+p[i]);
        }
        else{
            k[i][w]=k[i-1][w];
            //take upper value
        }
    }
    }
    cout<<"Total profit= "<<k[n][M]<<endl;

    //to know which obj is included or not-
    int i=n;
    int j=M;
    while(i>0 && j>0){
    //if that value is present in above row also then dont include it
    if(k[i][j]==k[i-1][j]){
        cout<<i<<"=0 ie not included  \n";
        i--;
    }
    else{
```

```
        cout<<i<<"=1 ie included \n";
        i--;
        j=j-wt[i];        //check weight after subtracting
    }
    }

//op- 12 and included- 1001
    return 0;
    }
```

OUTPUT:

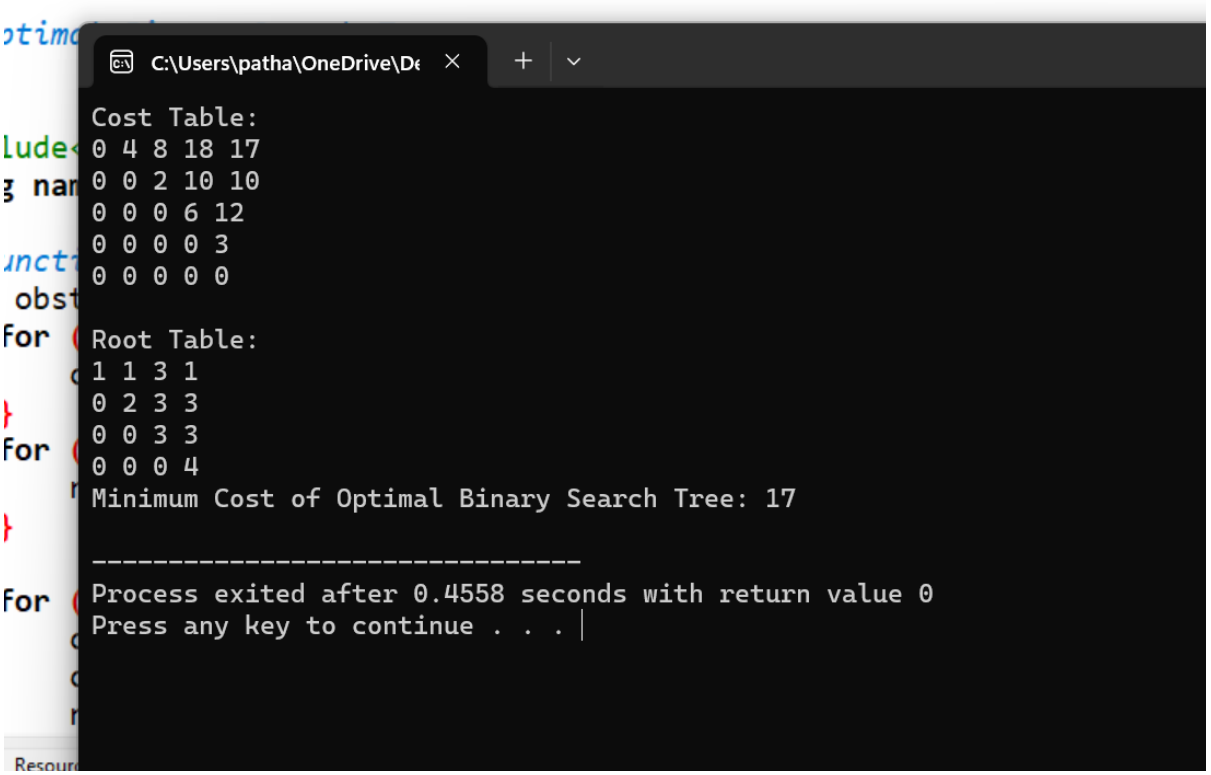


- **Optimal Binary Search Tree**

```
/*
4. Optimal Binary Search Tree
*/

#include<iostream>
using namespace std;

// Function to calculate optimal binary search tree
void obst(int n, int *keys, int *p, int **c, int **r) {
    for (int i = 0; i <= n + 1; i++) {
        c[i] = new int[n + 1]();
    }
    for (int i = 0; i <= n; i++) {
        r[i] = new int[n + 1]();
    }

    for (int i = 1; i <= n; i++) {
        c[i][i - 1] = 0;
        c[i][i] = p[i - 1];
        r[i][i] = i;
    }
    c[n + 1][n] = 0;

    for (int d = 1; d < n; d++) {
        for (int i = 1; i <= n - d; i++) {
            int j = d + i;
            int min = INT_MAX;

            for (int R = i; R <= j; R++) {
                int cost = c[i][R - 1] + c[R + 1][j];
                if (cost < min) {
                    min = cost;
                    r[i][j] = R;
                }
            }
            c[i][j] = min + p[j - 1] + p[i - 1];
        }
    }
}

int main() {
    // n- no of keys
    // p- probability of searching key
    int n = 4;
    int keys[] = {10, 20, 30, 40};
    int p[] = {4, 2, 6, 3};
    int **c = new int*[n + 2];
    int **r = new int*[n + 1];

    obst(n, keys, p, c, r);

    cout << "Cost Table: " << endl;
    for (int i = 1; i <= n + 1; i++) {
        for (int j = 0; j <= n; j++) {
            cout << c[i][j] << " ";
```

```
    }
    cout << endl;
}

cout << "\nRoot Table:" << endl;
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
        cout << r[i][j] << " ";
    }
    cout << endl;
}

cout << "Minimum Cost of Optimal Binary Search Tree: " << c[1][n] << endl;


return 0;
 }
```

OUTPUT:



- **Coin Exchange Problem**

```
/*
5. Coin Exchange Problem
*/

#include<iostream>
#include<algorithm>
using namespace std;

int min(int a,int b){
    return a<b? a:b;
}

void no_of_coins(int coins[],int w,int n){
    int i;          //coins array
    int j;          //amount into subproblems

    int a[n+1][w+1];

    for(i=0;i<=n;i++){
    for(j=0;j<=w;j++){
        a[i][j]=INT_MAX;            //initialise array to 0
    }
    }

    for(i=0;i<=n;i++){
    a[i][0]=0;              //for 0 amount we require no coins so op=0
    }

    for(j=1;j<=w;j++){
    a[0][j]=INT_MAX;            //if no coins available then no amount
    can be formed
    }

    for(i=1;i<=n;i++){
    for(j=1;j<=w;j++){
        //for case1 when coin>w
        if(coins[i-1]>j){
            //copy value from above row
            a[i][j]=a[i-1][j];
```

```
            }
            else{
                    a[i][j]=min(a[i-1][j],1+a[i][j-coins[i-1]]);
            }
    }
    }

    cout<<endl<<"Printing array "<<endl;
    for(int i=0;i<=n;i++){
    for(int j=0;j<=w;j++){
            cout<<a[i][j]<<" ";
    }
    cout<<endl;
    }

    cout<<endl<<"The minimum no of coins reqd to make sum of
    "<<w<<" is: "<<a[n][w]<<endl;

    //to find denomination::
    cout<<"The denomination of coins reqd to make sum "<<w<<" is:
    "<<endl;

    i=n;
    j=w;
    while(i>0 && j>0){
    if(a[i][j]!=a[i-1][j]){
            cout<<"included coin no: "<<i<<" with denomination:
    "<<coins[i-1]<<endl;
            j=j-coins[i-1];
    }
    else{
            i--;
    }
    }
}

int main()
{
    int n;
```

```
    cout<<"Enter total no of coins u have: \n";
    cin>>n;

//  int n=4;

    int coins[n];
    for(int i=0;i<n;i++){
    cout<<"Enter the coin no "<<i<<" : ";
    cin>>coins[i];
    }

//  int coins[]={1,5,6,9};


    int w;
    cout<<"Enter total amount u want to make: \n";
    cin>>w;

//  int w=10;

    no_of_coins(coins,w,n);



    return 0;
    }
```

OUTPUT: