EXPERIMENT-10

Program:

Implementation of the Longest Common Subsequence Problem using Dynamic Programming and calculate its complexity.

Pseudo Code:

```
function LCS(X[1..m], Y[1..n])
    let L[0..m, 0..n] be a 2D array
    for i := 0 to m
        for j := 0 to n
            if i = 0 or j = 0
                L[i, j] := 0
            else if X[i-1] = Y[j-1]
                L[i, j] := L[i-1, j-1] + 1
            else
                L[i, j] := max(L[i-1, j], L[i, j-1])
    return L[m, n]
```

Input:

```
#include<stdio.h>
#include<string.h>
 int i,j,m,n,c[20][20];
char x[20],y[20],b[20][20];
 void print(int i,int j)
{
            if(i==0 || j==0)
                        return;
            if(b[i][j]=='c')
            {
                        print(i-1,j-1);
                        printf("%c",x[i-1]);
            }
            else if(b[i][j]=='u')
```

```
                print(i-1,j);
        else
                print(i,j-1);
}
 void lcs()
{
        m=strlen(x);
        n=strlen(y);
        for(i=0;i<=m;i++)
                c[i][0]=0;
        for(i=0;i<=n;i++)
                c[0][i]=0;

        //c, u and l denotes cross, upward and downward directions respectively
        for(i=1;i<=m;i++)
                for(j=1;j<=n;j++)
                {
                        if(x[i-1]==y[j-1])
                        {
                                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';
```

```
                    }
                }
}
int main()
{
   printf("\nBoddu Asmitha Bhavya_A2305221386\n");
        printf("Enter 1st sequence:");
        scanf("%s",x);
        printf("Enter 2nd sequence:");
        scanf("%s",y);
        printf("\nThe Longest Common Subsequence is ");
        lcs();
        print(m,n);
return 0;
}
```

Output:

```
Boddu Asmitha Bhavya_A2305221386
Enter 1st sequence:asmitha
Enter 2nd sequence:ashi28a
The Longest Common Subsequence is asia
```

Complexity:

O(m*n)

Where m=length of first string

N=length of second string

EXPERIMENT-11

Program:

WAP to implement the Knapsack 0/1 problem using Backtracking and write the complexity.

Pseudo Code:

BK_KNAPSACK(M, W, V, fw, fp, X)

// Description : Solve knapsack problem using backtracking

// Input :

M: Knapsack capacity

W(1...n): Set of weight of the items

V(1...n): Set of profits associated with items

Fw: Final knapsack weight

Fp: Final earned profit

X(1...n): Solution vector

N: Total number of items

// Output : Solution tuple X, earned profit fp

// Initialization

cw ← 0 // Current weight

cp ← 0 // Current profit

fp ← – 1

k ← 1 // Index of item being processed

Input:

```
#include <stdio.h>
#include <stdlib.h>
int max(int a, int b) {
   return (a > b) ? a : b;
}
void knapsack(int W, int wt[], int val[], int n, int cur_wt, int cur_val, int *max_val, int sol[]) {
   if (cur_wt > W) {
      return;
   }
   if (n == 0) {
```

```
        if (cur_val > *max_val) {
            *max_val = cur_val;
            for (int i = 0; i < n; i++) {
                sol[i] = wt[i];
            }
        }
        return;
    }
    else {
        sol[n - 1] = 1;
        knapsack(W, wt, val, n - 1, cur_wt + wt[n - 1], cur_val + val[n - 1], max_val, sol);
        sol[n - 1] = 0;
        knapsack(W, wt, val, n - 1, cur_wt, cur_val, max_val, sol);
    }

}
int main() {
    int W;
    printf("\nBoddu Asmitha Bhavya_A2305221386\n");
    printf("Please enter the maximum weight that the knapsack can hold: ");
    scanf("%d", &W);
    int n;
    printf("Please enter the number of items: ");
    scanf("%d", &n);
   int wt[n], val[n];
    for (int i = 0; i < n; i++) {
        printf("Please enter the weight and value of item %d: ", i + 1);
        scanf("%d %d", &wt[i], &val[i]);
    }
    int max_val = 0;
    int sol[n];
```

```
    knapsack(W, wt, val, n, 0, 0, &max_val, sol);
    printf("The maximum value that can be obtained is %d\n", max_val);
    printf("The solution vector is: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", sol[i]);
    }
}
```

Output:'

```
Boddu Asmitha Bhavya_A2305221386
Please enter the maximum weight that the knapsack can hold: 8
Please enter the number of items: 4
Please enter the weight and value of item 1: 2 3
Please enter the weight and value of item 2: 3 5
Please enter the weight and value of item 3: 4 6
Please enter the weight and value of item 4: 5 10
The maximum value that can be obtained is 15
```

Complexity:

O(n*w)

Where n=number of items

W=the capacity of the Knapsack bag.