Sardar Patel College of Engineering, Bakrol

Experiment List

Name : Vishwas R. Acharya	**Semester : 5th**
Enrollment No : 181240116001	**Department : Information Technology**
Subject: Analysis and Design of Algorithm	**Subject Code : 2150703**

Sr. No	AIM	Experiment Date	Submission Date	Signature
1	Implementation and Time analysis of bubble sort.			
2	Implementation and Time analysis of selection sort.			
3	Implementation and Time analysis of insertion sort.			
4	Implementation and Time analysis of merge sort.			
5	Implementation and Time analysis of quick sort.			
6	Implementation of Binary Search			
7	Implementation and Time analysis of heap sort			
8	1 - Find the factorial of the given number using recursive function.			
	2 - Find the Fibonacci series using recursive function.			
9	Implementation of making change problem using dynamic.			
10	Implementation of a knapsack problem using dynamic programming.			
11	Implementation of chain matrix multiplication using dynamic programming			
12	Implementation of Prim's algorithm			
13	Implementation of Kruskal's algorithm			
14	Implementation of a knapsack problem using greedy programming.			
15	Implementation of Graph and Searching (DFS).			
16	Implementation of Graph and Searching (BFS).			
17	Implement LCS problem.			

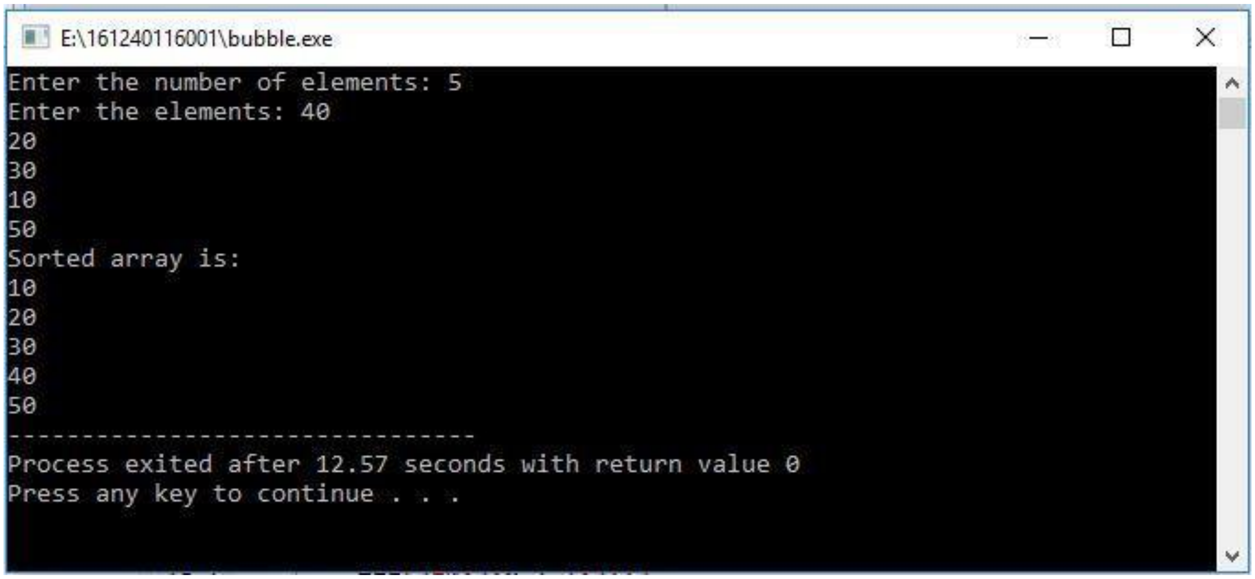
Practical – 1

AIM: Implementation and Time analysis of bubble sort.

Solution:

```
#include<iostream>

using      namespace

std; int main()

{

int a[100],n,i,j,swap;       cout<<"Enter the

number of elements: ";

        cin>>n;

cout<<"Enter the elements: ";   for(i=0;i<n;i++)cin>>a[i];       for(i=0;i<n;i++){


        for(j=0;j<n-i-1;j++){

        if(a[j]>a[j+1]){

swap=a[j];

a[j]=a[j+1];

a[j+1]=swap;

                        }

                }

        }

cout<<"Sorted array is: ";

for(i=0;i<n;i++){

cout<<endl<<a[i];

        }
        return 0;

}
```

Output:



```
Enter the number of elements: 5
Enter the elements: 40
20
30
10
50
Sorted array is:
10
20
30
40
50
--------------------------------
Process exited after 12.57 seconds with return value 0
Press any key to continue . . .
```

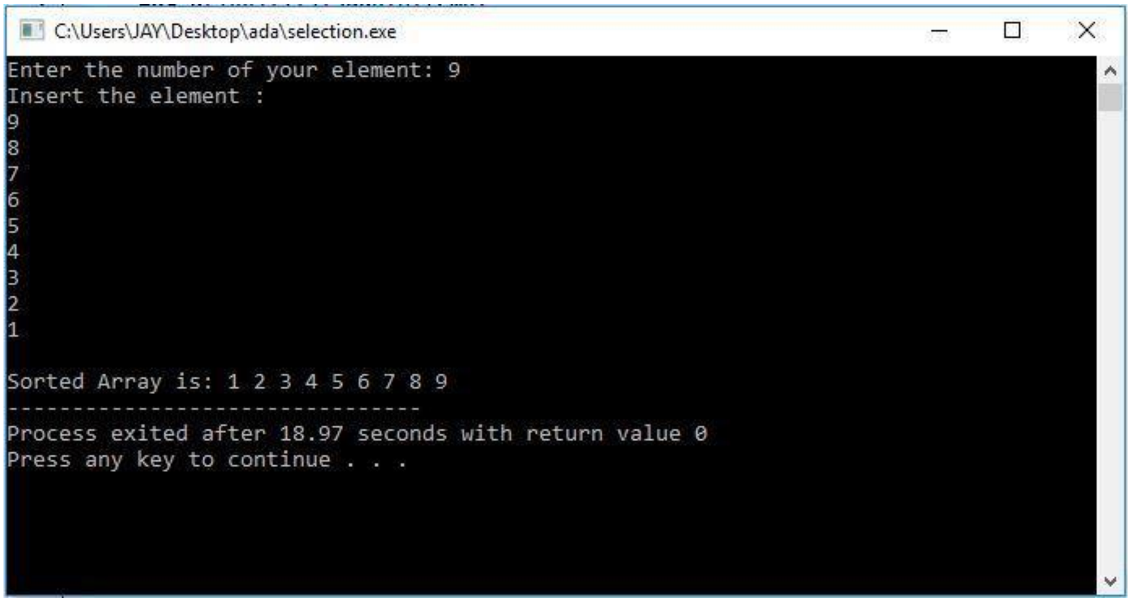
Practical – 2

AIM: Implementation and Time analysis of selection sort.

Solution:

```
#include<iostream>
using       namespace
std; int main()
{
int a[100],i,j,swap,n,temp;         cout<<"Enter the
number of your element: ";
          cin>>n;
cout<<"Insert the element :\n";
for(i=0;i<n;i++)
          {
                    cin>>a[i];
          }
          for(i=0;i<n;i++)
          {
                    swap=i;
                    for(j=i+1;j<n;j++)
                    {
                              if(a[swap]>a[j])
                                        swap=j;
          temp=a[i];
a[i]=a[swap];
a[swap]=temp;
                    }
          }
cout<<endl<<"Sorted Array is:";
for(i=0;i<n;i++)
{ cout<<" "<<a[i];
          }
          return 0;
}
```

Output:



```
Enter the number of your element: 9
Insert the element :
9
8
7
6
5
4
3
2
1

Sorted Array is: 1 2 3 4 5 6 7 8 9
--------------------------------
Process exited after 18.97 seconds with return value 0
Press any key to continue . . .
```

Practical – 3

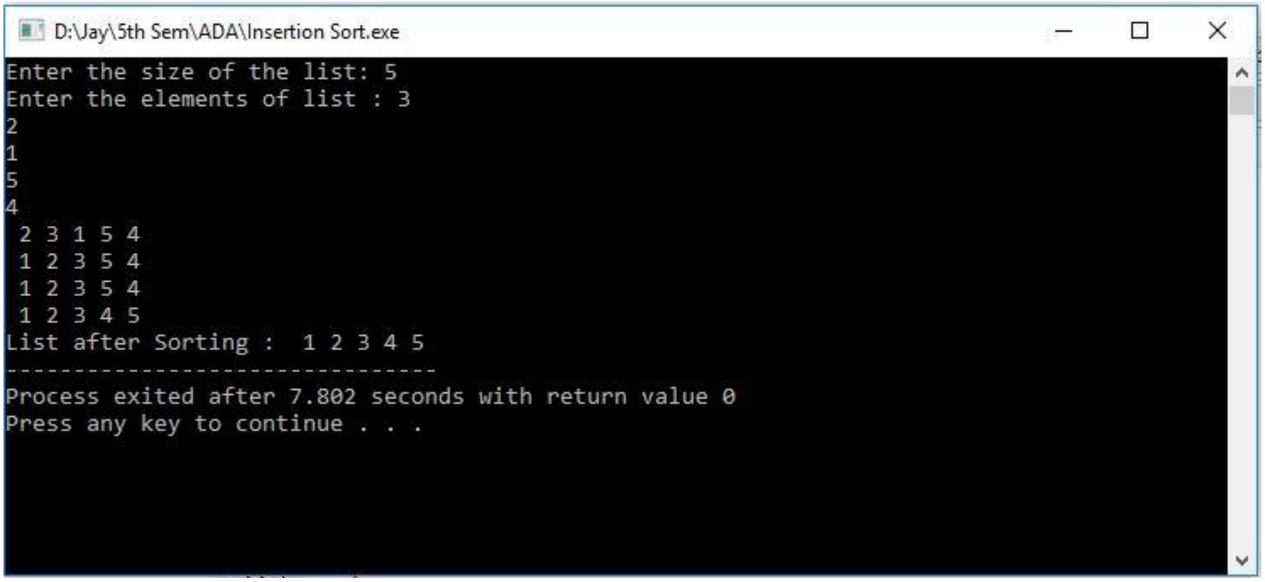
AIM: Implementation and Time analysis of insertion sort.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int size, i, j, temp, a[100];
cout<<"Enter the size of the list: ";
cin>>size;
cout<<"Enter the elements of list : ";
for (i = 0; i< size; i++)
{cin>>a[i];}
for (i = 1; i< size; i++)
{
temp = a[i];
 j = i - 1;
while ((temp < a[j]) && (j >= 0))
        {
        a[j + 1] = a[j];
         j = j - 1;
}
a[j + 1] = temp;
```

```
 for(j=0;j<size;j++)
{
          cout<<" "<<a[j];
          }cout<<endl;
}
cout<<"List after Sorting : ";
for (i = 0; i< size; i++)
{
          cout<<" "<<a[i];
}
return 0;
}
```

Output:



D:\Jay\5th Sem\ADA\Insertion Sort.exe

```
Enter the size of the list: 5
Enter the elements of list : 3
2
1
5
4
 2 3 1 5 4
 1 2 3 5 4
 1 2 3 5 4
 1 2 3 4 5
List after Sorting :  1 2 3 4 5
--------------------------------
Process exited after 7.802 seconds with return value 0
Press any key to continue . . .
```

Practical – 4

AIM: Implementation and Time analysis of merge sort.

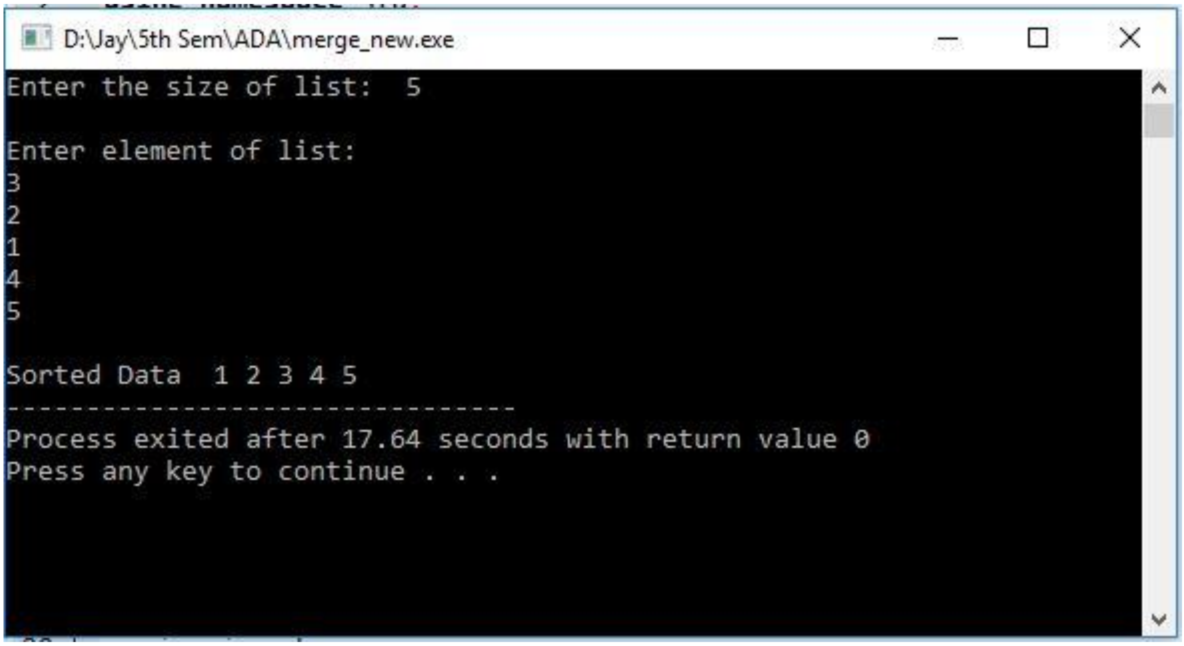
Solution:

```
#include <iostream>

using namespace std;

int Merge(int *a, int low, int high, int mid){

        int i, j, k, temp[high-low+1];

        i = low;

        k = 0;

        j = mid + 1;

while (i<= mid && j <= high){

                if (a[i] < a[j]){

                        temp[k] = a[i];

                        k++;

                        i++;

                }

                else{

                        temp[k] = a[j];

                        k++;

                        j++;

                }

        }
```

```
while (i<= mid){
        temp[k] = a[i];
        k++;
        i++;
    }
while (j <= high){
        temp[k] = a[j];
        k++;
        j++;
    }
for (i = low; i<= high; i++){
        a[i] = temp[i-low];
    }
}
int MergeSort(int *a, int low, int high){
    int mid;
    if (low <high){
        mid=(low+high)/2;
        MergeSort(a, low, mid);
        MergeSort(a, mid+1, high);
        Merge(a, low, high, mid);
    }
```

```
}
int main(){
    int n, i;
    cout<<"Enter the size of list:  ";
    cin>>n;
int arr[n];
cout<<"\nEnter element of list:  "<<endl;
    for(i = 0; i< n; i++){
        cin>>arr[i];
    }
MergeSort(arr, 0, n-1);
cout<<"\nSorted Data ";
    for (i = 0; i< n; i++){
    cout<<" "<<arr[i];
    }
return 0;
}
```

Output:



```
Enter the size of list:  5

Enter element of list:
3
2
1
4
5

Sorted Data  1 2 3 4 5
--------------------------------
Process exited after 17.64 seconds with return value 0
Press any key to continue . . .
```

Practical – 5

AIM: Implementation and Time analysis of quick sort.

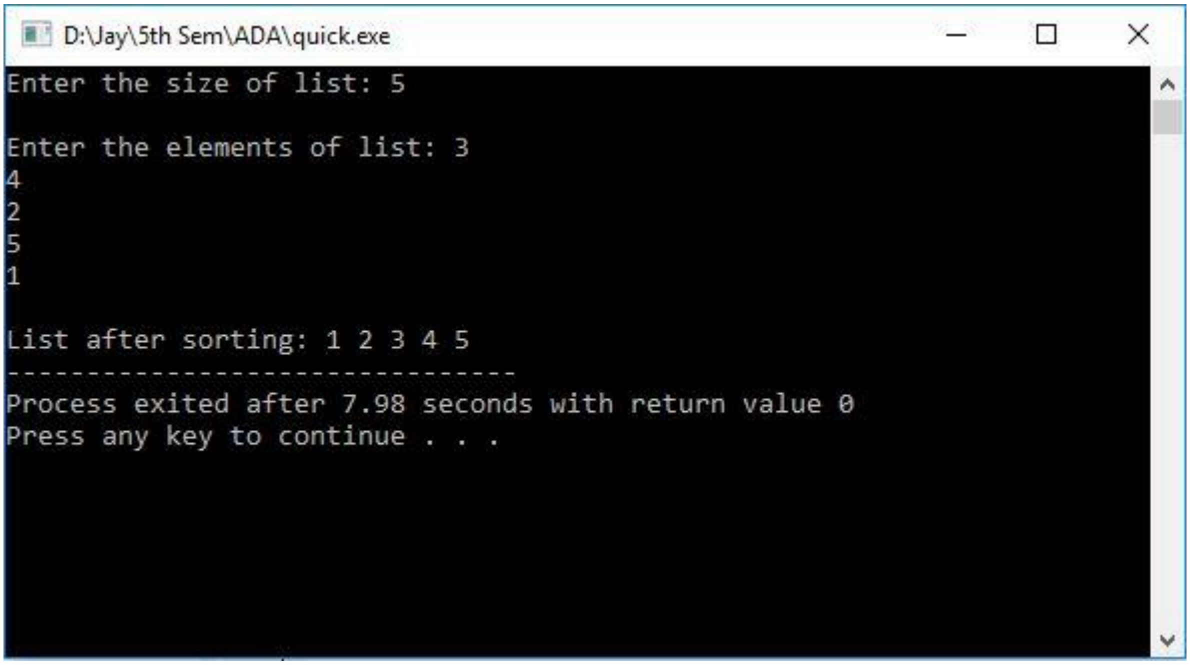
Solution:

```
#include <iostream>
using namespace std;
void quick_sort(int[],int,int);
int partition(int[],int,int);
int main(){
    int a[50],n,i;
cout<<"Enter the size of list: ";
cin>>n;
cout<<"\nEnter the elements of list: ";
    for(i=0;i<n;i++){
cin>>a[i];
     }
quick_sort(a,0,n-1);
cout<<"\nList after sorting: ";
    for(i=0;i<n;i++){
          cout<<a[i]<<" ";
     }
    return 0;
}
 void quick_sort(int a[],int l,int u){
    int j;
```

```
   if(l<u){
      j=partition(a,l,u);
quick_sort(a,l,j-1);
quick_sort(a,j+1,u);
     }
}
 int partition(int a[],int l,int u){
   int v,i,j,temp;
   v=a[l];
i=l;
   j=u+1;
do{
do{
        i++;
}while(a[i]<v&&i<=u);
do{
        j--;
}while(v<a[j]);
       if(i<j){
         temp=a[i];
         a[i]=a[j];
         a[j]=temp;
      }
}while(i<j);
```

```
    a[l]=a[j];

    a[j]=v;

    return(j);

}
```

Output:



D:\Jay\5th Sem\ADA\quick.exe

```
Enter the size of list: 5

Enter the elements of list: 3
4
2
5
1

List after sorting: 1 2 3 4 5
--------------------------------
Process exited after 7.98 seconds with return value 0
Press any key to continue . . .
```

Practical – 6

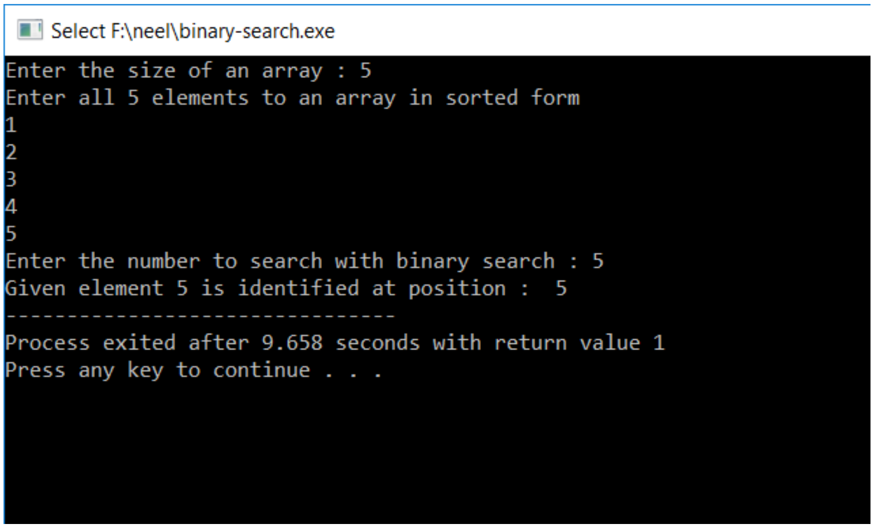
AIM: Implementation of Binary Search

Solution:

```
#include<iostream>
using namespace std;
int arr[50];
int search(int low,int high, int a){
 int mid=(low+high)/2, count;
        if(a == arr[mid]){
        count=0;
        cout<<"Given element "<<a<<" is identified at position :  "<<mid+1;
        return 0;
}
else if(a <arr[mid]){
        int high1 = mid;
        search(low, high1, a);
}
else if(a >arr[mid]){
        int low1 = mid+1;
        search(low1,high,a);
}
return 1;
}
```

```
int main(){
        int n,zero=0,count=0;
        cout<<"Enter the size of an array : ";
        cin>>n;
        int val;
        cout<<"Enter all "<<n<<" elements to an array in sorted form \n";
        for(int k=0; k<n; k++)cin>>arr[k];
        cout<<"Enter the number to search with binary search : ";
        cin>>val;
        search(zero,n,val);
        return 1;
}
```

Output:



```
Enter the size of an array : 5
Enter all 5 elements to an array in sorted form
1
2
3
4
5
Enter the number to search with binary search : 5
Given element 5 is identified at position :  5
--------------------------------
Process exited after 9.658 seconds with return value 1
Press any key to continue . . .
```

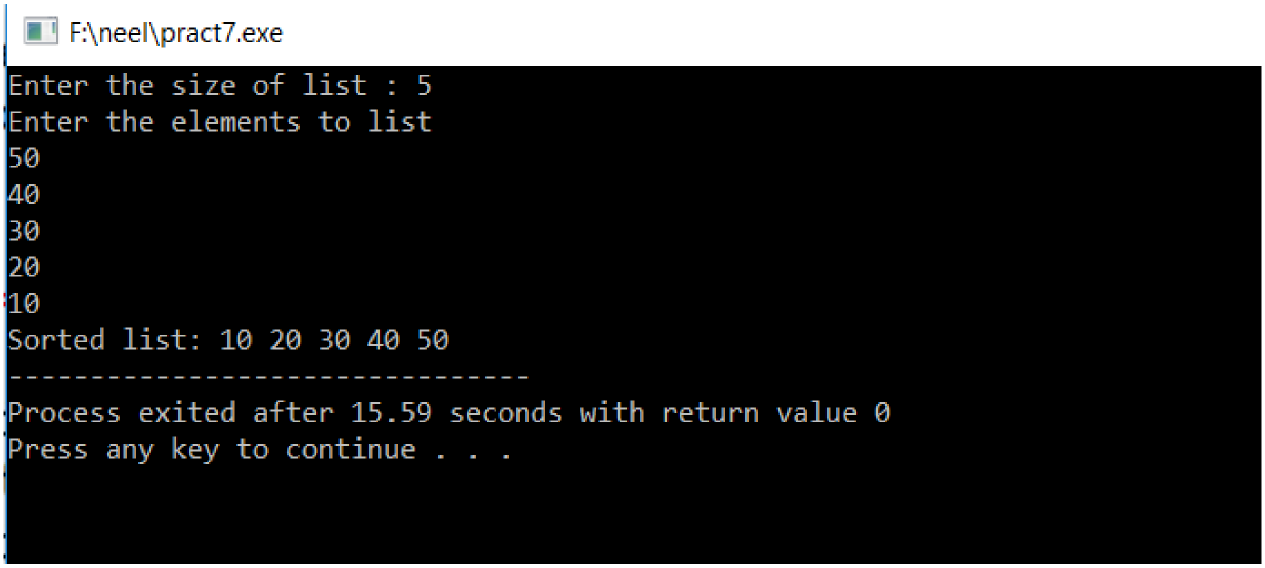
Practical – 7

AIM: Implementation and Time analysis of heap sort.

Solution:

```
#include <iostream>
using namespace std;
void buildHeap(int array[],int size,int i)
{
    int max = i;
    int left = 2*i+1;
    int right = 2*i+2;
    if(left<size && array[left]>array[max])
          max = left;
     if(right<size && array[right]>array[max])
          max = right;
     if(max!=i)
          {
                    swap(array[i],array[max]);
                    buildH
          eap(array,size
          ,max);
          }
}
void heapSort(int array[],int size)
{
    for(int i=size/2-1;i>=0;--i)
    buildHeap(array,size,i);
    for(int i=size-1;i>=0;i--)
    {
        swap(array[0],array[i]);
        buildHeap(array,i,0);
    }
}
int main()
{
    int size;
    cout<<"Enter the size of list : ";
    cin>>size;
    int array[size],n=0;
    cout<<"Enter the elements to list \n";
    while(n<size)cin>>array[n++];
    heapSort(array,size);
    cout<<"Sorted list: ";
    for(int i=0;i<size;i++)
    cout<<array[i]<<" ";
    return 0;
}
```

Output:



F:\neel\pract7.exe

```
Enter the size of list : 5
Enter the elements to list
50
40
30
20
10
Sorted list: 10 20 30 40 50
--------------------------------
Process exited after 15.59 seconds with return value 0
Press any key to continue . . .
```

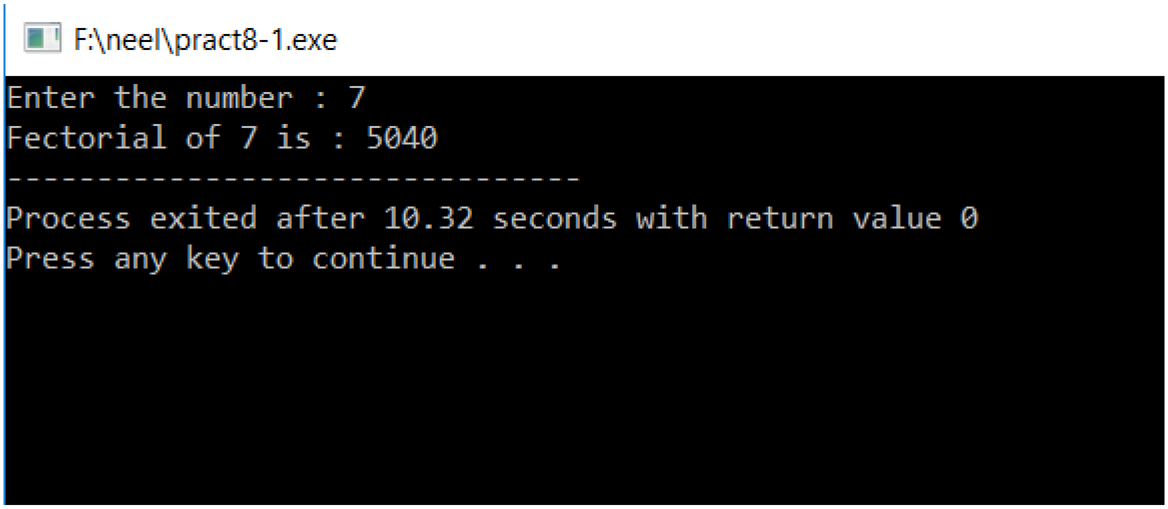
Practical – 8.1

AIM: Find the factorial of the given number using recursive function.

Solution:

```
#include<iostream>
using namespace std;
int fact(int n)
{
	if(n!=0)
	{
		return n * fact(n-1);
	}
	else
	{
		return 1;
	}
}
int main()
{
	int data;
	cout<<"Enter the number : ";
	cin>>data;
	cout<<"Fectorial of "<<data<<" is : "<<fact(data);
}
```

Output:



```
Enter the number : 7
Fectorial of 7 is : 5040
--------------------------------
Process exited after 10.32 seconds with return value 0
Press any key to continue . . .
```

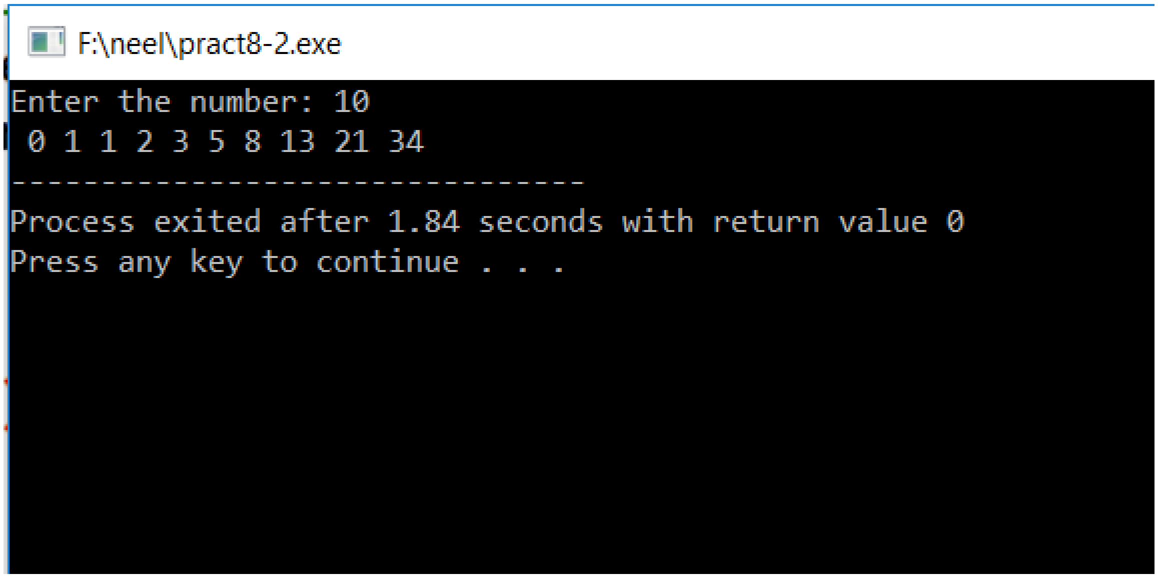
Practical – 8.2

AIM: Find the Fibonacci series using recursive function.

Solution:

```
#include<iostream>
using namespace std;
int feb(int n)
{
        if((n==1)||(n==0))
        {
                return (n);
        }
        else
        {
                return (feb(n-1)+feb(n-2));
        }
}
int main()
{
        int number,i=0;
        cout<<"Enter the number: ";
        cin>>number;
        while(i<number)
        {
                cout<<" "<<feb(i);
                i++;
        }
        return 0;
}
```

Output:



```
Enter the number: 10
 0 1 1 2 3 5 8 13 21 34
--------------------------------
Process exited after 1.84 seconds with return value 0
Press any key to continue . . .
```

Practical – 9

AIM: Implementation of making change problem using dynamic programming.

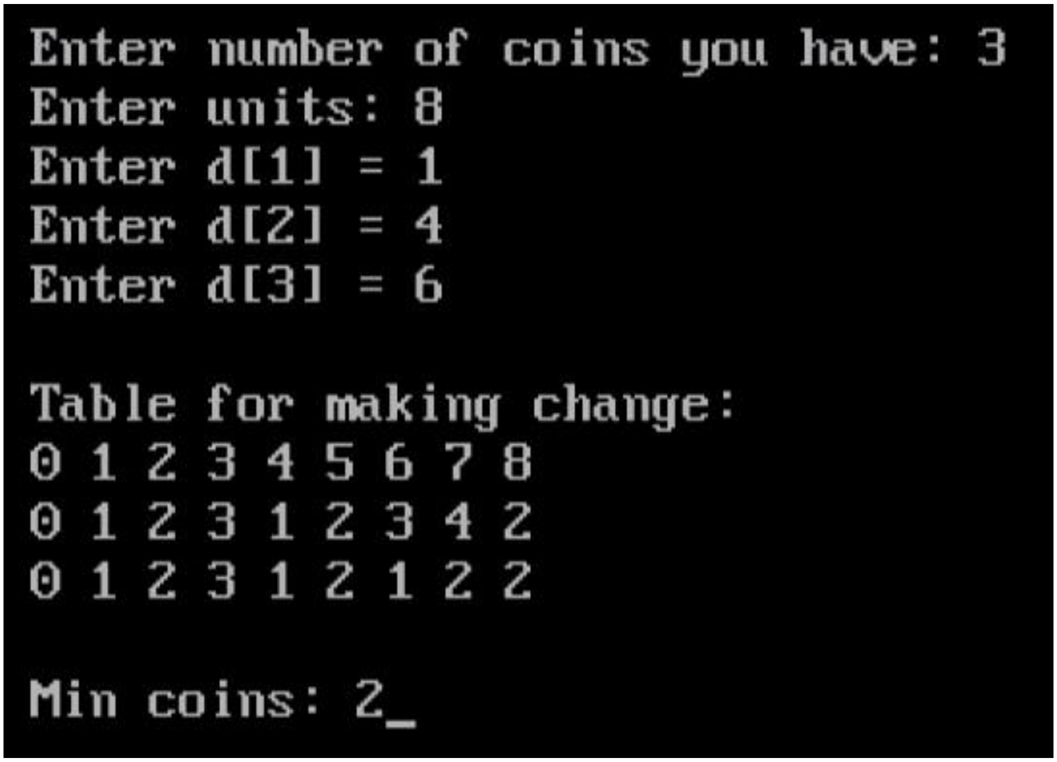
Solution:

```
#include<stdio.h>

#include<conio.h>

void main()

{

int d[100],mk[100][100],n,N,i=0,j=0,a,b;

clrscr();

printf("Enter number of coins you have: ");

scanf("%d",&n);

printf("Enter units: ");

scanf("%d",&N);

for(i=1;i<=n;i++)

{

        printf("Enter d[%d] = ",i);

        scanf("%d",&d[i]);

}

for(i=1;i<=n;i++)

{

        for(j=0;j<=N;j++)

        {

                if(j==0)
```

```
                        mk[i][0]=0;
                else if(i==1)
                        mk[1][j]=1+mk[1][j-d[i]];
                else if(j<d[i])
                        mk[i][j]=mk[i-1][j];
                else
                {
                        a=mk[i-1][j];
                        b=1+mk[i][j-d[i]];
                        if(a<b)
                        {
                                mk[i][j]=a;
                        }
                        else
                        {
                                mk[i][j]=b;
                        }
                }
        }
}
printf("\nTable for making change:\n");

for(i=1;i<=n;i++){
        for(j=0;j<=N;j++)
```

```
                printf("%d ",mk[i][j]);
        printf("\n");
}
printf("\nMin coins: %d",mk[n][N]);
getch();
}
```

Output:



```
Enter number of coins you have: 3
Enter units: 8
Enter d[1] = 1
Enter d[2] = 4
Enter d[3] = 6

Table for making change:
0 1 2 3 4 5 6 7 8
0 1 2 3 1 2 3 4 2
0 1 2 3 1 2 1 2 2

Min coins: 2_
```

Practical - 10

AIM: Implementation of a knapsack problem using dynamic programming.

Solution:-

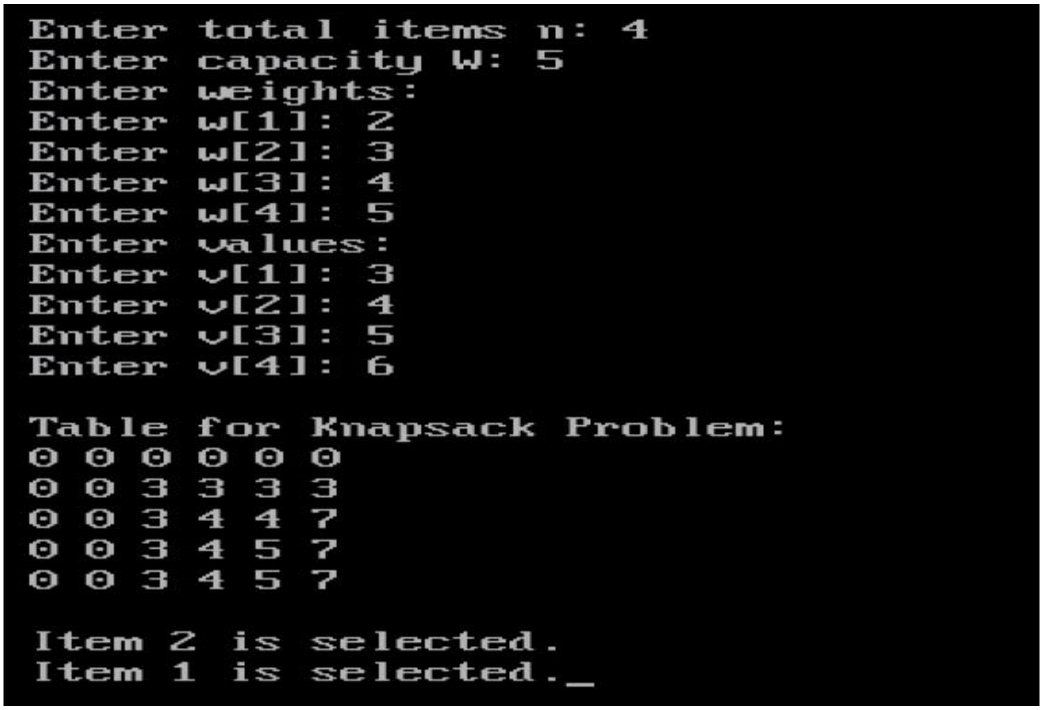
```
#include<stdio.h>
#include<conio.h>
void main()
{
intn,W,w[100],v[100],t[100][100],i=0,j=0,a,b;
clrscr();
printf("Enter total items n: ");
scanf("%d",&n);
printf("Enter capacity W: ");
scanf("%d",&W);
printf("Enter weights: \n");
for(i=0;i<=n;i++)
{
        printf("Enter w[%d]: ",i);
        scanf("%d",&w[i]);
}
printf("Enter values: \n");
for(i=0;i<=n;i++)
{
        printf("Enter v[%d]: ",i);
        scanf("%d",&v[i]);
```

```
}


        for(i=0;i<=n;i++)
{
        for(j=0;j<=W;j++)
        {
                if(i==0 || j==0)
                {
                        t[i][j]=0;
                }
                else if(j<w[i])
                {
                        t[i][j]=t[i-1][j];
                }
                else
                {
                        a=t[i-1][j];
                        b=v[i]+t[i-1][j-w[i]];
                        if(a>b)
                        {
                                t[i][j]=a;
                        }
                        else
                        {
```

```
                t[i][j]=b;
            }
        }
    }
}
printf("\nTable for Knapsack Problem:\n");
for(i=0;i<=n;i++)
{
    for(j=0;j<=W;j++)
    {

            printf("%d ",t[i][j]);
    }
    printf("\n");
}
j=W;
for(i=n;i>0;i--)
{
        if(t[i][j]!=t[i-1][j])
        {
            printf("\nItem %d is selected.",i);
            j=j-w[i];
        }
}
```

```
getch();

}
```

Output:



```
Enter total items n: 4
Enter capacity W: 5
Enter weights:
Enter w[1]: 2
Enter w[2]: 3
Enter w[3]: 4
Enter w[4]: 5
Enter values:
Enter v[1]: 3
Enter v[2]: 4
Enter v[3]: 5
Enter v[4]: 6

Table for Knapsack Problem:
0 0 0 0 0 0
0 0 3 3 3 3
0 0 3 4 4 7
0 0 3 4 5 7
0 0 3 4 5 7

 Item 2 is selected.
 Item 1 is selected._
```

Practical – 11

Aim: Implementation of chain matrix multiplication using dynamic programming.

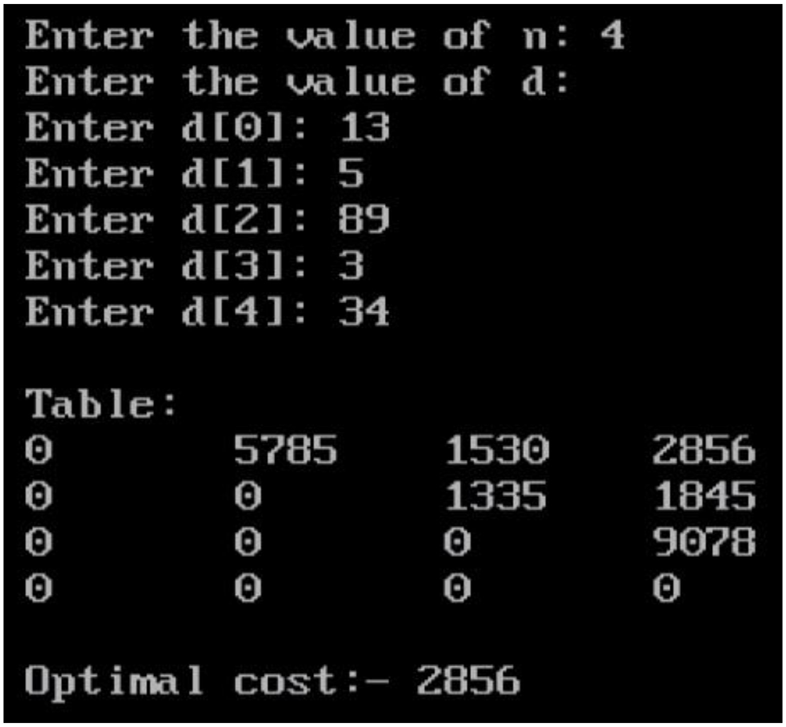
Program Input:

```
#include<stdio.h>

#include<conio.h>

void main(){

int d[100],m[100][100],n,i=0,j=0,k=0,s=0,t[10],l=0,temp;

clrscr();

printf("\nEnter the value of n: ");

scanf("%d",&n);

printf("\nEnter the value of d:");

for(i=0;i<=n;i++){

        printf("\nEnter d[%d]: ",i);

        scanf("%d",&d[i]);

}

for(s=0;s<n;s++){

        if(s==0){

                for(i=1;i<=n;i++){

                        m[i][i]=0;

                }

        }

        else if(s==1){
```

```
        for(i=1;i<n;i++)
        {
            m[i][i+1]=(d[i-1]*d[i]*d[i+1]);
        }
    }
    else
    {
        for(i=1;i<=(n-s);i++)
        {
            l=0;
            for(k=i;k<(i+s);k++)
            {
                t[l++]=m[i][k]+m[k+1][i+s]+(d[i-1]*d[k]*d[i+s]);
            }
            for(k=1;k<l;k++){
                temp=t[0];
                if(t[k]<temp){
                    temp=t[k];
                }
            }
            m[i][i+s]=temp;
        }
    }
}
```

```
printf("\nTable:\n");
for(i=1;i<=n;i++){
        for(j=1;j<=n;j++)
                printf("%d\t",m[i][j]);
        printf("\n");
}
printf("\nOptimal cost:- %d",m[1][n]);
getch();
}
```

Output:



```
Enter the value of n: 4
Enter the value of d:
Enter d[0]: 13
Enter d[1]: 5
Enter d[2]: 89
Enter d[3]: 3
Enter d[4]: 34

Table:
0       5785    1530    2856
0       0       1335    1845
0       0       0       9078
0       0       0       0

Optimal cost:- 2856
```

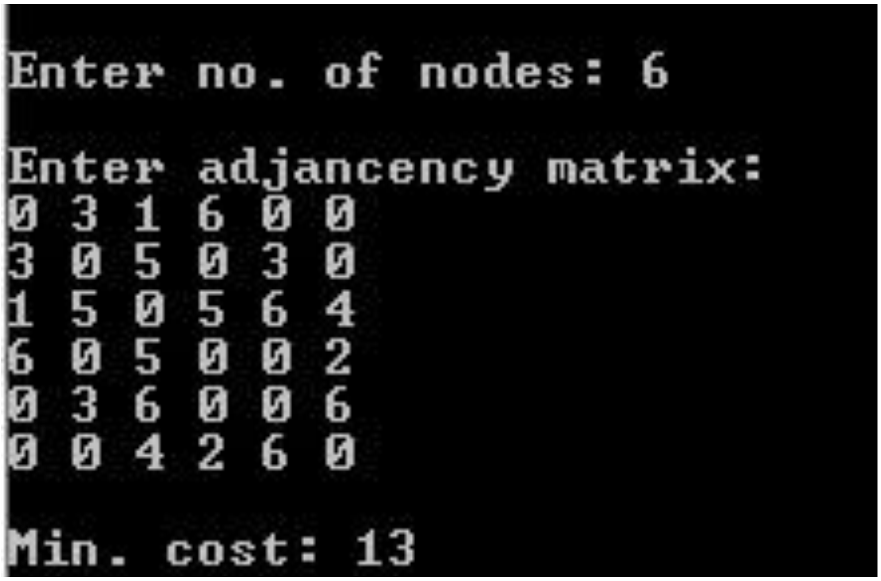
Practical - 12

AIM: Implement prim's algorithm.

Solution:-

```
#include<stdio.h>
#include<conio.h>
int main()
{
int am[100][100],i=0,j=0,n,min=0,mc=0,a;
printf("\nEnter no. of nodes: ");
scanf("%d",&n);
printf("\nEnteradjancency matrix: \n");
for(i=1;i<=n;i++)
   {
for(j=1;j<=n;j++)
      {
scanf("%d",&am[i][j]);
      }
   }
for(i=1;i<n;i++)
   {
min=999;
for(j=1;j<=n;j++){
        if(am[i][j]!=0 && am[i][j]<min){
                min=am[i][j];
           a=j;
                }
        }
mc=mc+min;
am[a][i]=0;
    }
printf("\nMin. cost: %d",mc);
return 0;
}
```

Output:



```
Enter no. of nodes: 6

Enter adjancency matrix:
0 3 1 6 0 0
3 0 5 0 3 0
1 5 0 5 6 4
6 0 5 0 0 2
0 3 6 0 0 6
0 0 4 2 6 0

Min. cost: 13
```

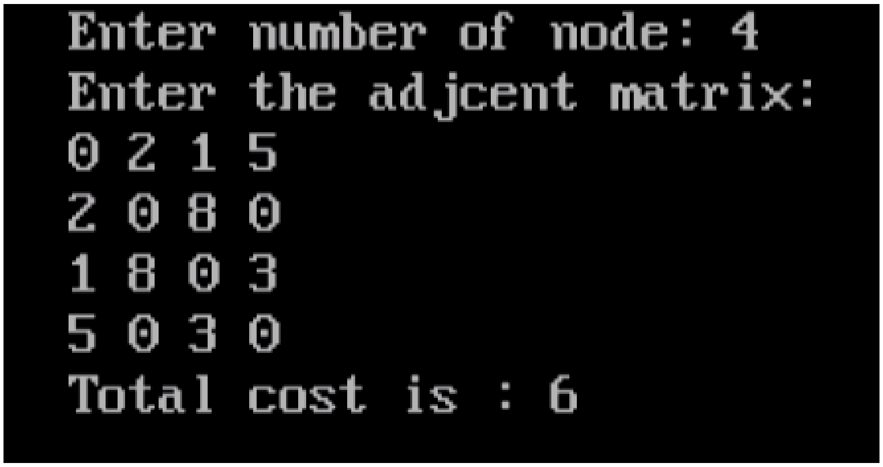
Practical -13

AIM: Implement Kruskal’s algorithm.

Solution:-

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[100][100],i,j,s=1,n,min=99,cost=0,x,y;
    printf("Enter number of node: ");
    scanf("%d",&n);
    printf("Enter the adjcent matrix: \n");
    for(i=1;i<=n;i++){
        for(j=1;j<=n;j++)
            scanf("%d",&a[i][j]);
    }
    while(s<n){
        for(i=1;i<=n;i++){
            for(j=1;j<=n;j++)  {
                if(j>i && a[i][j]!=0){
                    if(a[i][j]<min){
                    min=a[i][j];
                    x=i;
                    y=j;
                }
            }
        }
        }
        cost=cost+min;
        s++;
        min=99;
                a[x][y]=0;
}
printf("Total cost is : %d",cost);
getch();
}
```

Output:



```
Enter number of node: 4
Enter the adjcent matrix:
0 2 1 5
2 0 8 0
1 8 0 3
5 0 3 0
Total cost is : 6
```

Practical -14

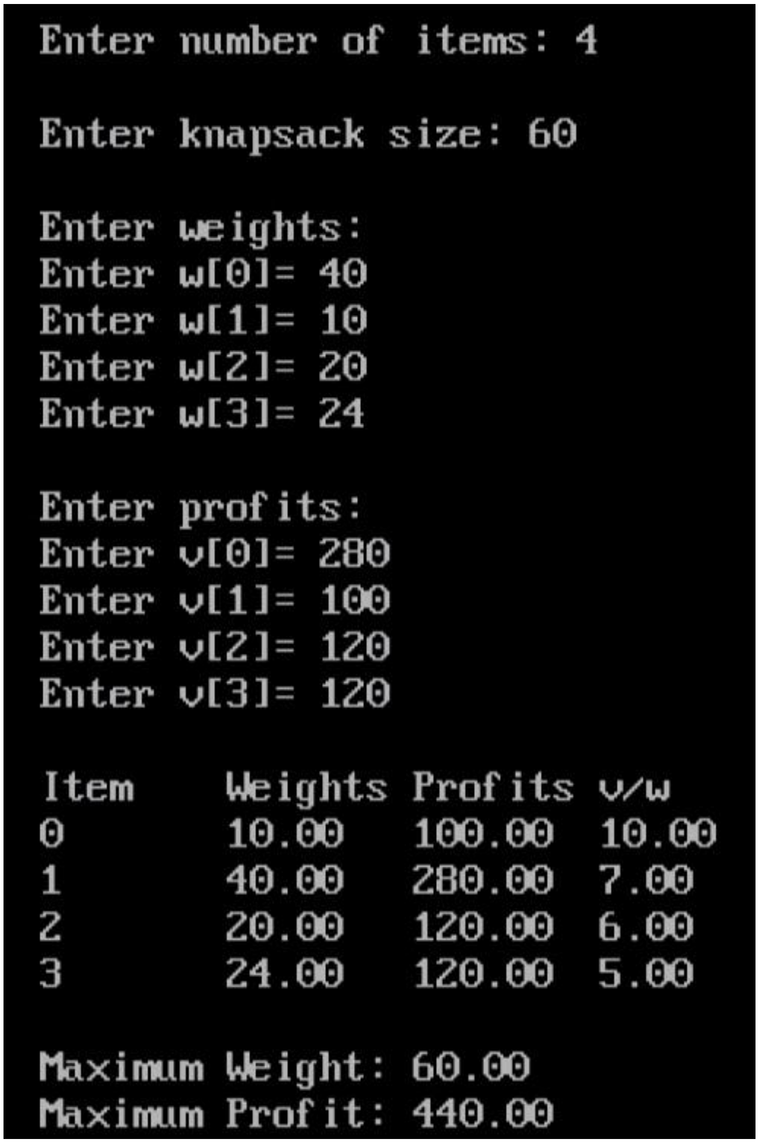
AIM: Implementation of a knapsack problem using greedy algorithm.

Solution:-

```
#include<stdio.h>
#include<conio.h>
void main()
{
intn,i=0,j=0,s;
float v[100],w[100],vw[100],t,f[50]={0},mw=0.0,mp=0.0;
printf("\nEnter number of items: ");
scanf("%d",&n);
printf("\nEnter knapsack size: ");
scanf("%d",&s);
printf("\nEnter weights: \n");
for(i=0;i<n;i++)
{
        printf("Enter w[%d]= ",i);
        scanf("%f",&w[i]);
}
printf("\nEnter profits: \n");
for(i=0;i<n;i++)
{
        printf("Enter v[%d]= ",i);
        scanf("%f",&v[i]);
}
```

```
for(i=0;i<n;i++)
{
        vw[i]=(v[i]/w[i]);
}
for(i=0;i<n;i++)
{
        for(j=0;j<=i;j++)
        {
                if(vw[i]>vw[j])
                {
                        t=vw[i];
                        vw[i]=vw[j];
                        vw[j]=t;
                        t=v[i];
                        v[i]=v[j];
                        v[j]=t;
                        t=w[i];
                        w[i]=w[j];
                        w[j]=t;
                }
        }
}
printf("\nItem\tWeights\tProfits\tv/w");
for(i=0;i<n;i++)
{
        printf("\n%d\t%.2f\t%.2f\t%.2f",i,w[i],v[i],vw[i]);
}
for(i=0;i<n;i++)
{
        if(w[i]>s)
                break;
        else
        {
                f[i]=1.0;
                s=s-w[i];
        }
}
if(i<n)
        f[i]=s/w[i];
for(i=0;i<n;i++){
        w[i]=w[i]*f[i];
        v[i]=v[i]*f[i];
}
for(i=0;i<n;i++){
        mw=mw+w[i];
        mp=mp+v[i];

}
printf("\n\nMaximum Weight: %.2f",mw);
printf("\nMaximum Profit: %.2f",mp);
getch();
}
```

Output:



```
Enter number of items: 4

Enter knapsack size: 60

Enter weights:
Enter w[0]= 40
Enter w[1]= 10
Enter w[2]= 20
Enter w[3]= 24

Enter profits:
Enter v[0]= 280
Enter v[1]= 100
Enter v[2]= 120
Enter v[3]= 120

Item    Weights Profits v/w
0       10.00   100.00  10.00
1       40.00   280.00  7.00
2       20.00   120.00  6.00
3       24.00   120.00  5.00

Maximum Weight: 60.00
Maximum Profit: 440.00
```

Practical -15

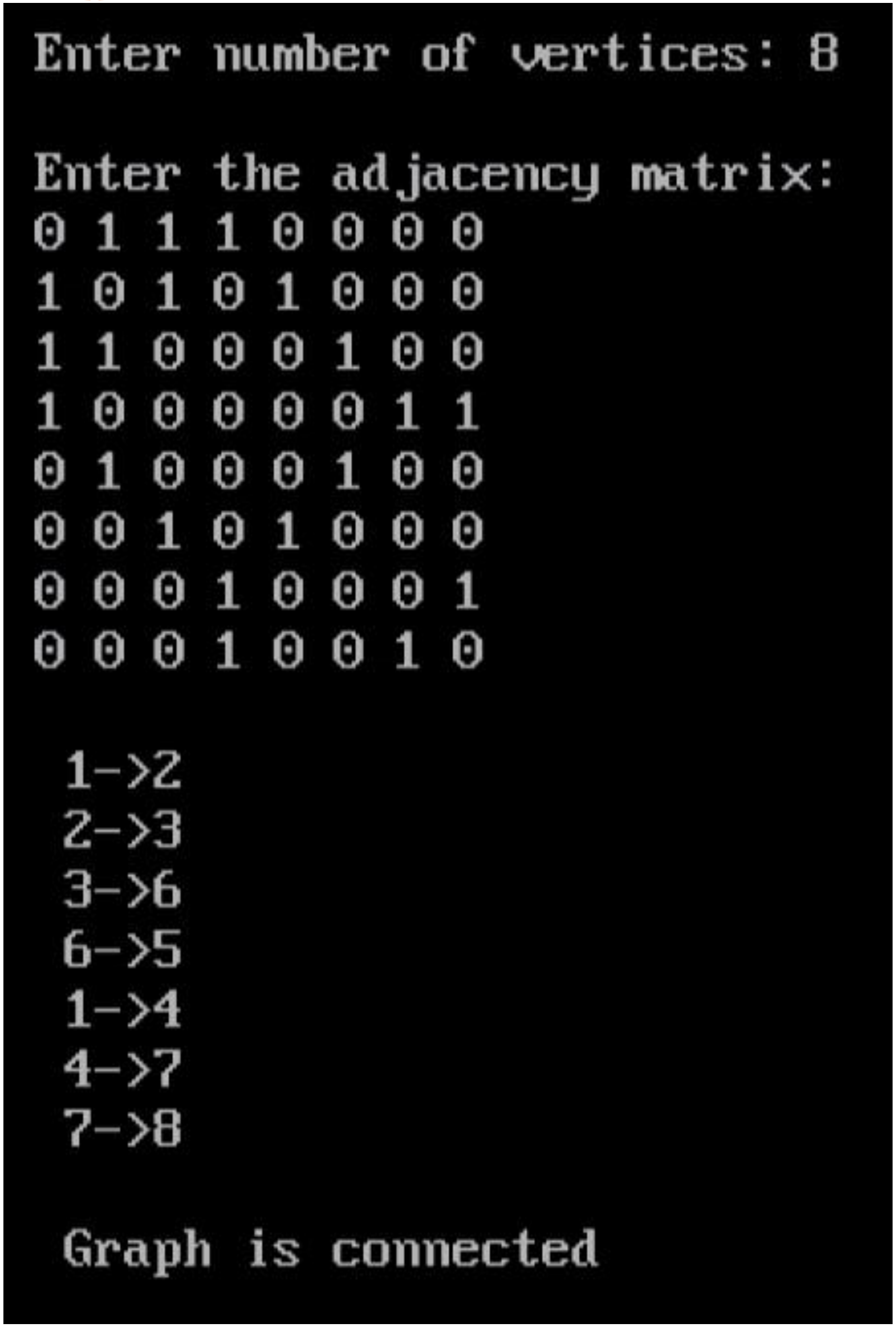
AIM: Implementation of Graph and Searching (DFS).

Solution:-

```
#include<stdio.h>
#include<conio.h>
int a[20][20],reach[20],n;
voiddfs(int v){
int i;
reach[v]=1;
for(i=1;i<=n;i++)   {
        if(a[v][i] && !reach[i]){
                printf("\n%d->%d",v,i);
                dfs(i);
        }
        }
}
void main(){
inti,j,count=0;
printf("\nEnter number of vertices: ");
scanf("%d",&n);
for(i=1;i<=n;i++){
        reach[i]=0;
        for(j=1;j<=n;j++)
        {
                a[i][j]=0;
```

```
          }
}
printf("\nEnter the adjacency matrix:\n");
for(i=1;i<=n;i++){
          for (j=1;j<=n;j++)
                    scanf("%d",&a[i][j]);
}
dfs(1);
printf("\n");
for(i=1;i<=n;i++){
          if(reach[i]){
                    count++;
          }
}
if(count==n)
{
          printf("\nGraph is connected");
}
else
{
          printf("\nGraph is not connected");
}
getch();
}
```

Output:



```
Enter number of vertices: 8

Enter the adjacency matrix:
0 1 1 1 0 0 0 0
1 0 1 0 1 0 0 0
1 1 0 0 0 1 0 0
1 0 0 0 0 0 1 1
0 1 0 0 0 1 0 0
0 0 1 0 1 0 0 0
0 0 0 1 0 0 0 1
0 0 0 1 0 0 1 0

 1->2
 2->3
 3->6
 6->5
 1->4
 4->7
 7->8

 Graph is connected
```

Practical -16

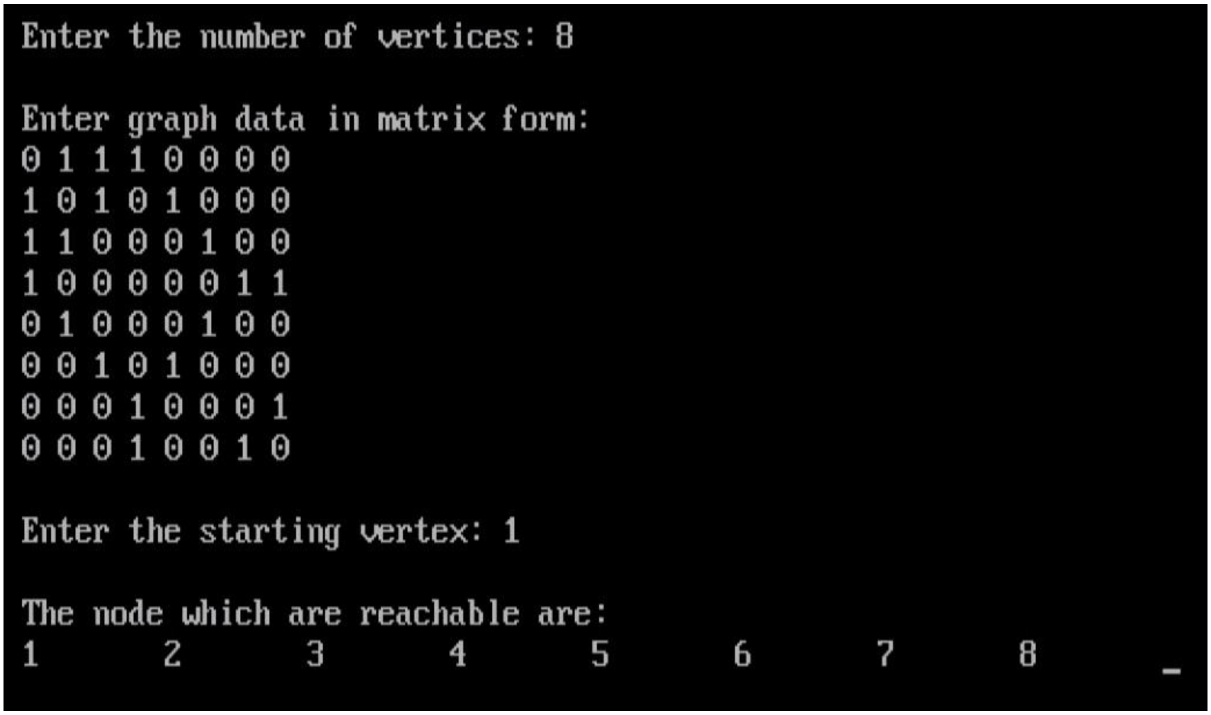
AIM: Implementation of Graph and Searching (BFS).

Solution:-

```
#include<stdio.h>
#include<conio.h>
int a[20][20],q[20],visited[20],n,i,j,f=0,r=-1;
voidbfs(int v)
{
for(i=1;i<=n;i++)
{
        if(a[v][i] && !visited[i])
        {
                q[++r]=i;
        }
}
if(f<=r)
{
        visited[q[f]]=1;
        bfs(q[f++]);
}
}
void main()
{
int v;
printf("\nEnter the number of vertices: ");
```

```
scanf("%d",&n);
for (i=1;i<=n;i++)

{
        q[i]=0;
        visited[i]=0;
}
printf("\nEnter graph data in matrix form:\n");
for (i=1;i<=n;i++)
{
        for (j=1;j<=n;j++)
        {
                scanf("%d",&a[i][j]);
        }
}
printf("\nEnter the starting vertex: ");
scanf("%d",&v);
bfs(v);
printf("\nThe node which are reachable are:\n");
for (i=1;i<=n;i++)
{
        if(visited[i])
        {
                printf("%d\t",i);
        }
        else
        {
                printf("\n Bfs is not possible");
        }
}
getch();
}
```

Output:



```
Enter the number of vertices: 8

Enter graph data in matrix form:
0 1 1 1 0 0 0 0
1 0 1 0 1 0 0 0
1 1 0 0 0 1 0 0
1 0 0 0 0 0 1 1
0 1 0 0 0 1 0 0
0 0 1 0 1 0 0 0
0 0 0 1 0 0 0 1
0 0 0 1 0 0 1 0

Enter the starting vertex: 1

The node which are reachable are:
1       2       3       4       5       6       7       8       _
```

Practical - 17

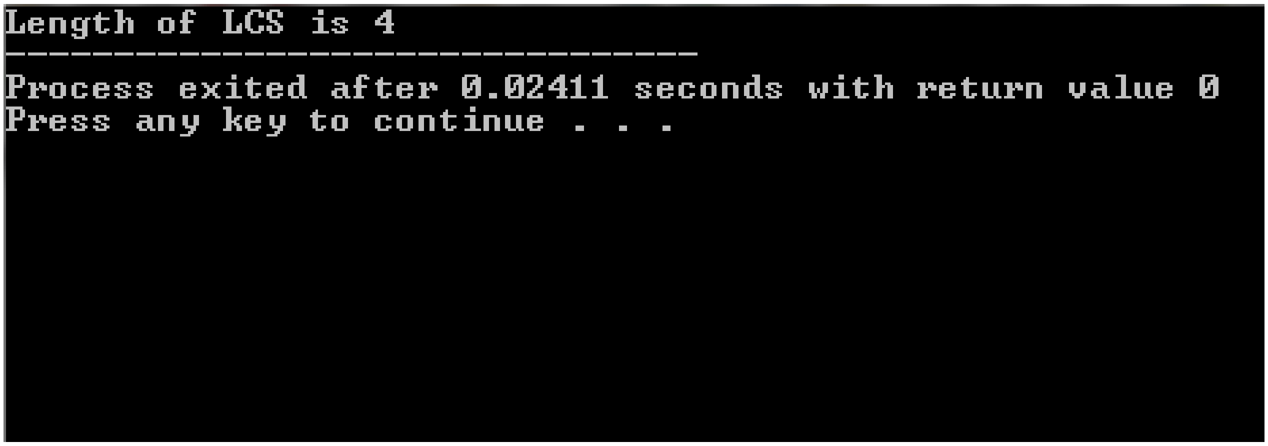
AIM: Implement LCS problem.

Solution:

```
#include<bits/stdc++.h>

int max(int a, int b);

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

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

   int i, j;

  for (i=0; i<=m; i++) {

     for (j=0; j<=n; j++) {

       if (i == 0 || 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];

}

int max(int a, int b) {

    return (a > b)? a : b;

}

int main()
```

```
{
  char X[] = "AGGTAB";
  char Y[] = "GXTXAYB";
  int m = strlen(X);
  int n = strlen(Y);
   printf("Length of LCS is %d", lcs( X, Y, m, n ) );
  return 0;
}
```

Output:



```
Length of LCS is 4
--------------------------------
Process exited after 0.02411 seconds with return value 0
Press any key to continue . . .
```