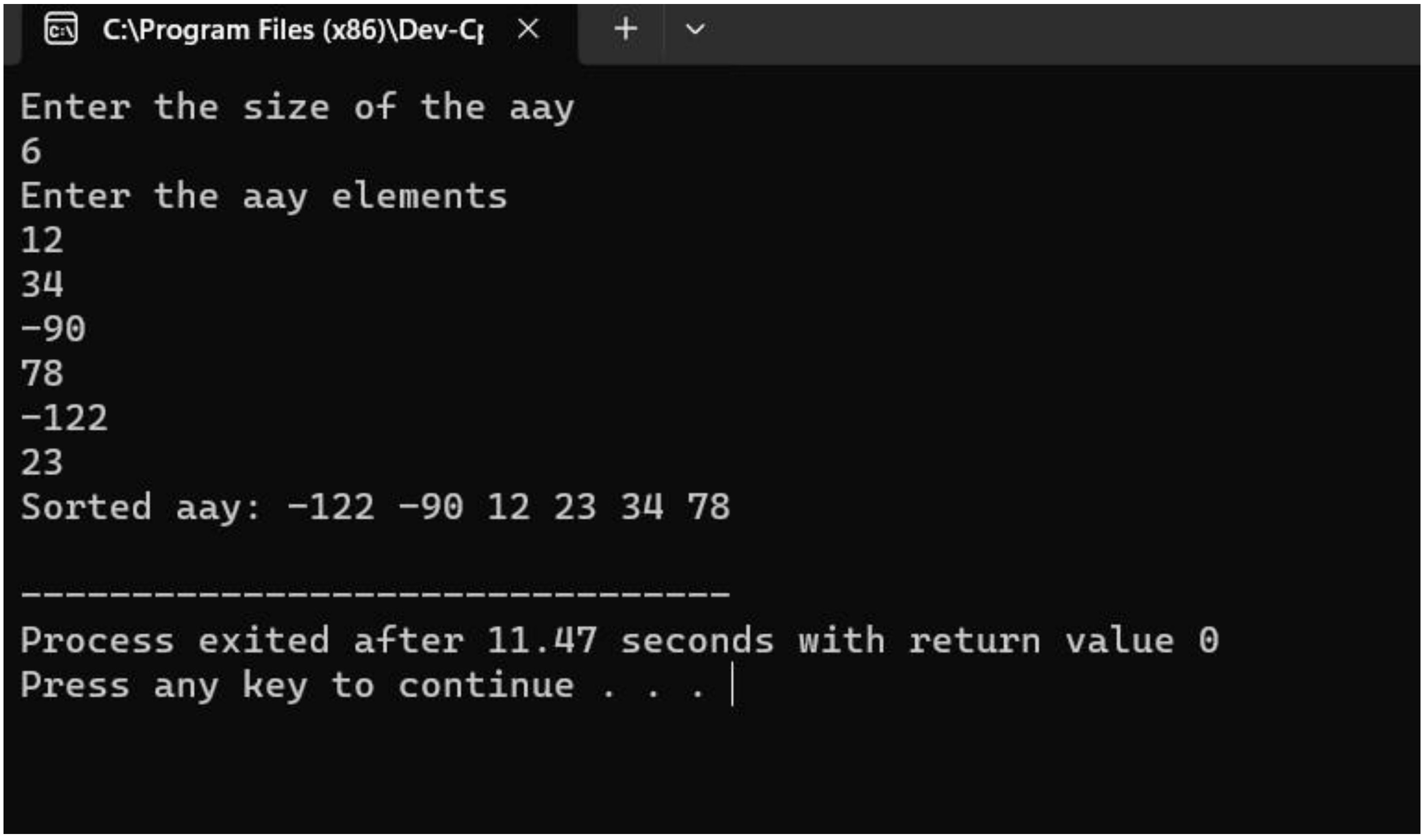
Part-A

1. Write a program to sort a list of N element using Selection Sort Technique.

```
#include <stdio.h>
int main()
 {
   int a[20],n,i;
   printf("Enter the size of the aay\n");
   scanf("%d",&n);
   printf("Enter the aay elements\n");
   for(i=0;i<n;i++)
   {
      scanf("%d",&a[i]);
   }
        for (int i = 0; i < n - 1; i++) {
      int min = i;
      for (int j = i + 1; j < n; j++)
                {
         if (a[j] < a[min])
                         {
            min= j;
         }
      }
      int temp = a[min];
      a[min] = a[i];
      a[i] = temp;
   }
   printf("Sorted aay: ");
   for (i = 0; i <n; i++)
        {
      printf("%d ", a[i]);
    }
   printf("\n");
   return 0;
}
```

Output:

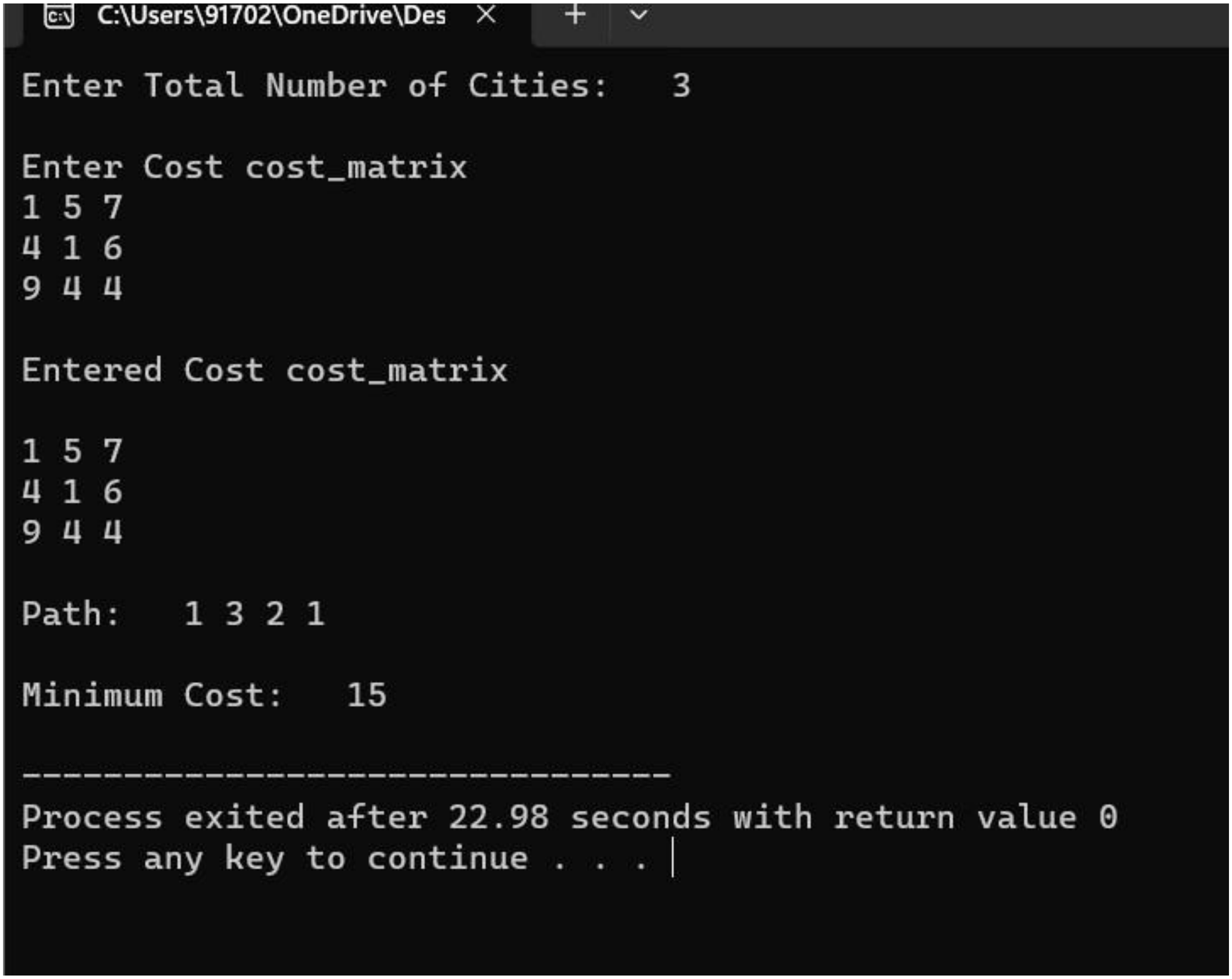


```
Enter the size of the aay
6
Enter the aay elements
12
34
-90
78
-122
23
Sorted aay: -122 -90 12 23 34 78

--------------------------------
Process exited after 11.47 seconds with return value 0
Press any key to continue . . .
```

2. Write a program to perform Travelling Salesman Problem.

```
#include<stdio.h>
int cost_matrix[25][25], visited[10], n, cost = 0;
int tsp(int v)
{
    int i, nearest_city = 999;
    int minimum = 999, temp;
    for(i = 0; i < n; i++)
    {
        if((cost_matrix[v][i] != 0) && (visited[i] == 0))
        {
            if(cost_matrix[v][i] < minimum)
            {
                minimum = cost_matrix[i][0] + cost_matrix[v][i];
            }
            temp = cost_matrix[v][i];
            nearest_city = i;
        }
    }
    if(minimum != 999)
    {
        cost = cost + temp;
    }
    return nearest_city;
}
void minimum_cost(int city)
{
    int nearest_city;
    visited[city] = 1;
    printf("%d ", city + 1);
    nearest_city = tsp(city);
    if(nearest_city == 999)
    {
        nearest_city = 0;
        printf("%d", nearest_city + 1);
        cost = cost + cost_matrix[city][nearest_city];
        return;
    }
    minimum_cost(nearest_city);
}
int main()
{
    int i, j;
    printf("Enter Total Number of Cities:\t");
    scanf("%d", &n);
    printf("\nEnter Cost cost_matrix\n");
    for(i = 0; i < n; i++)
    {
        for(j = 0; j < n; j++)
        {
            scanf("%d", &cost_matrix[i][j]);
        }
    }
    for(i = 0; i < n; i++)
    {
    visited[i] = 0;
   }
    printf("\nEntered Cost cost_matrix\n");
    for(i = 0; i < n; i++)
    {
        printf("\n");
        for(j = 0; j < n; j++)
        {
            printf("%d ", cost_matrix[i][j]);
        }
    }
    printf("\n\nPath:\t");
    minimum_cost(0);
    printf("\n\nMinimum Cost: \t");
    printf("%d\n", cost);
    return 0;
}
```

Output:

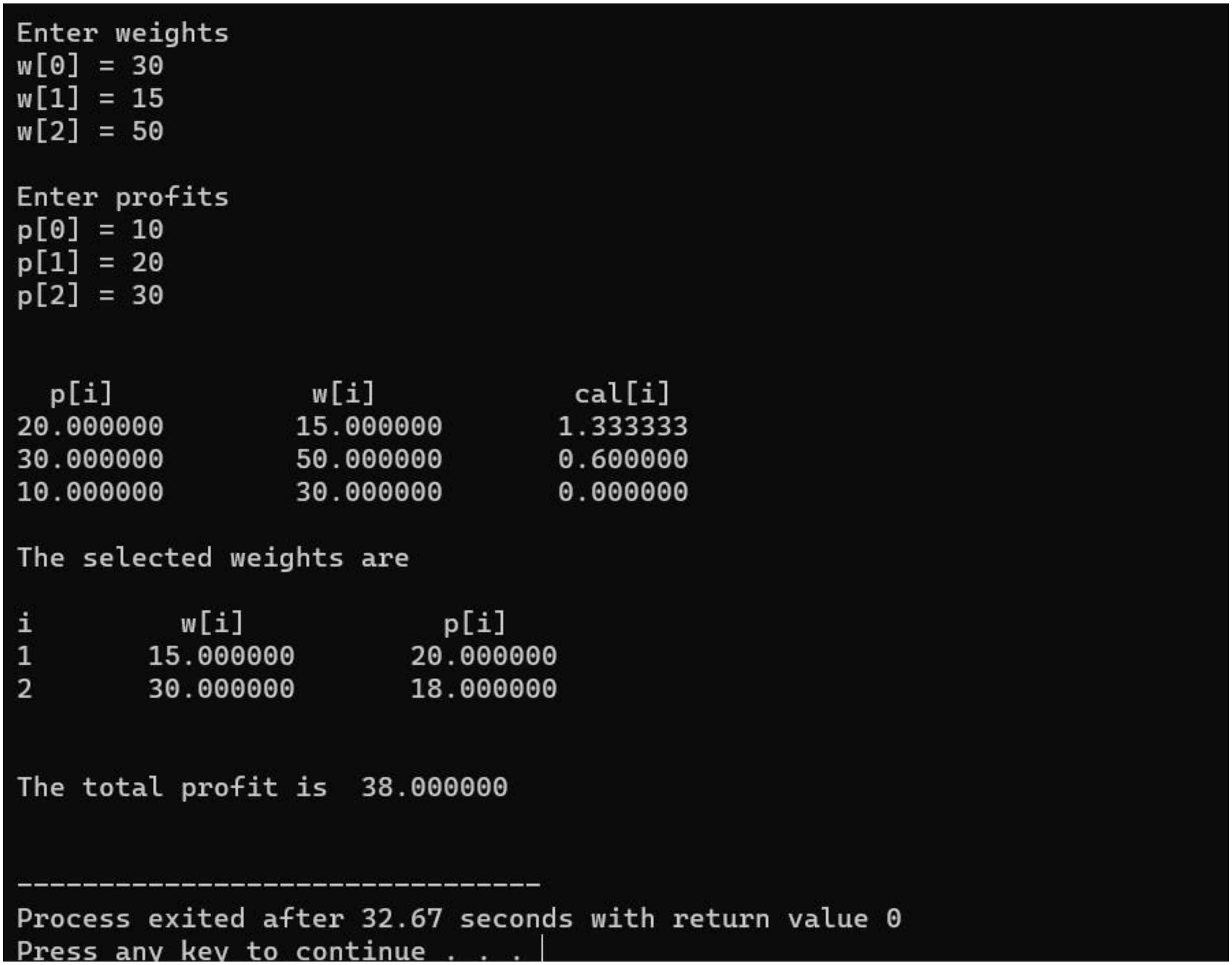


```
Enter Total Number of Cities:   3

Enter Cost cost_matrix
1 5 7
4 1 6
9 4 4

Entered Cost cost_matrix

1 5 7
4 1 6
9 4 4

Path:   1 3 2 1

Minimum Cost:   15

--------------------------------
Process exited after 22.98 seconds with return value 0
Press any key to continue . . .
```

3. Write a program to perform Knapsack Problem using Greedy Solution.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int m, n, i, j;
    printf("Enter maximum weight of knapsack    ");
    scanf("%d", &m);
    printf("\nEnter number of objects    ");
    scanf("%d", &n);
    int wt = 0, k = 0;
    float cal[n], p[n], w[n], x[n], prof = 0;
    for (i = 0; i < n; i++)
        x[i] = 0;
    printf("\nEnter weights\n");
    for (i = 0; i < n; i++)
    {
        printf("w[%d] = ", i);
        scanf("%f", &w[i]);
    }
    printf("\nEnter profits\n");
    for (i = 0; i < n; i++)
    {
        printf("p[%d] = ", i);
        scanf("%f", &p[i]);
    }
    for (i = 0; i < n; i++)
        cal[i] = p[i] / w[i];
    for (i = 0; i < n; i++)
    {
        for (j = i + 1; j < n; j++)
        {
            if (cal[i] < cal[j])
            {
                int t1, t2, t3;
                t1 = cal[i];
                cal[i] = cal[j];
                cal[j] = t1;
                t2 = w[i];
                w[i] = w[j];
                w[j] = t2;
                t3 = p[i];
                p[i] = p[j];
                p[j] = t3;
            }
        }
    }
    printf("\n\n  p[i]\t\t  w[i]\t\t  cal[i]\n");
    for (i = 0; i < n; i++)
        printf("%f\t %f\t %f\t\n", p[i], w[i], cal[i]);
    for (i = 0; i < n; i++)
    {
        if ((wt + w[i]) <= m)
        {
            k++;
            x[i] = 1;
            wt += w[i];
            prof += p[i];
        }
        else
        {
            k++;
            x[i] = (m - wt) / w[i];
            w[i] = m - wt;
            wt = m;
            prof += (x[i] * p[i]);
            p[i] = (x[i] * p[i]);
            break;
        }
        printf("\nThe selected weights are \n\ni\t  w[i]\t\t  p[i]\n");
        for (i = 0; i < k; i++)
            printf("%d\t%f\t%f\n", i + 1, w[i], p[i]);
        printf("\n\nThe total profit is  %f\n\n", prof);
        return 0;
    }
```

Output:

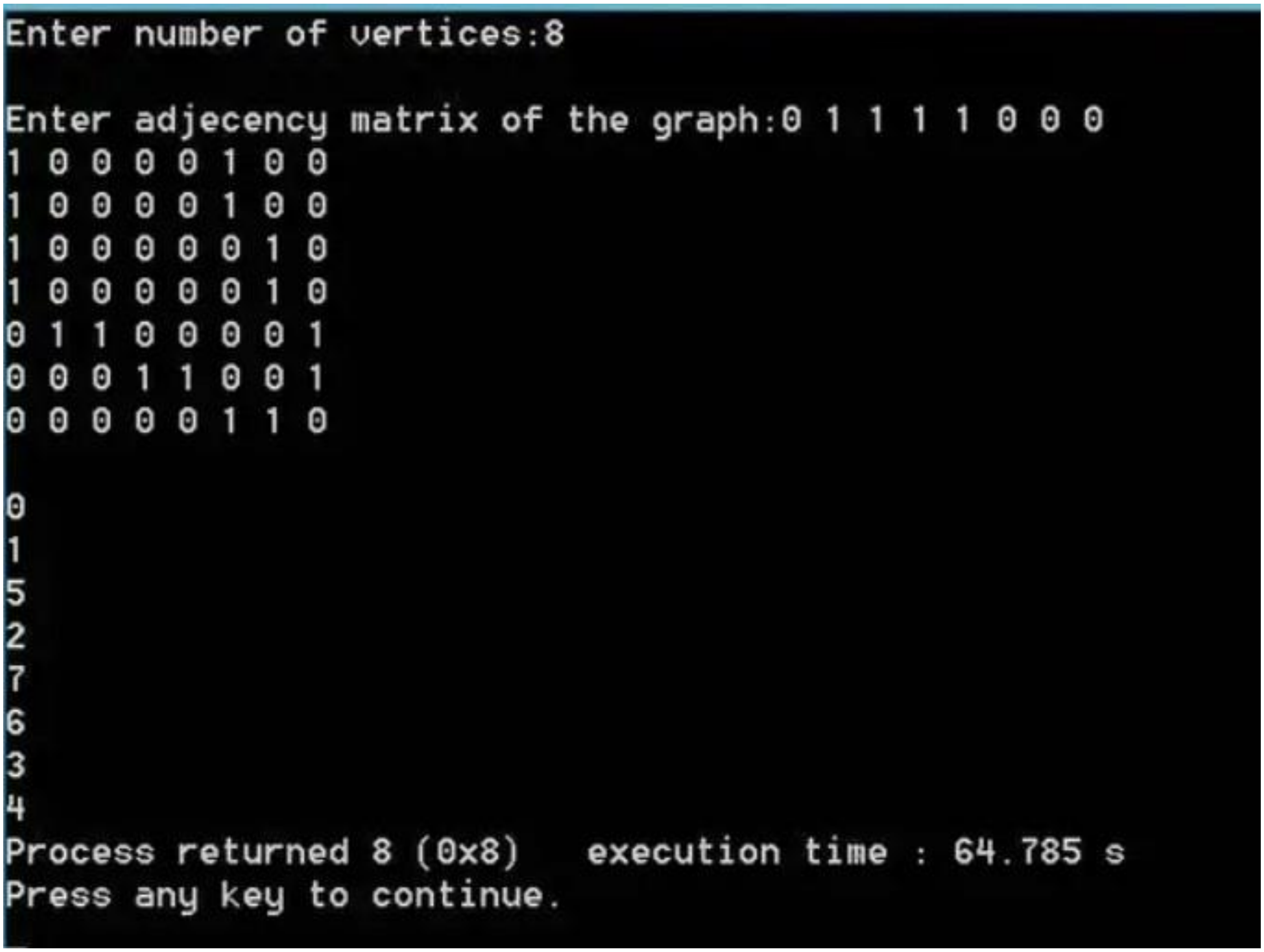


```
Enter weights
w[0] = 30
w[1] = 15
w[2] = 50

Enter profits
p[0] = 10
p[1] = 20
p[2] = 30


  p[i]            w[i]            cal[i]
20.000000        15.000000       1.333333
30.000000        50.000000       0.600000
10.000000        30.000000       0.000000

The selected weights are

i         w[i]            p[i]
1       15.000000       20.000000
2       30.000000       18.000000


The total profit is  38.000000


--------------------------------
Process exited after 32.67 seconds with return value 0
Press any key to continue . . .
```

4. Write a program to implement the DFS algorithm for a graph.

```
#include<stdio.h>
int a[10][10],visited[10],n;
void DFS(int v)
{
    int k;
printf("%d->",v);
    visited[v]=1;
for(k=0;k<n;k++)
      if(visited[k]==0 && a[v][k]==1)
      {
             DFS(k);
       }
}
int main()
{
    int i,j,v;
    printf("Enter number of vertices:");
  scanf("%d",&n);
  printf("\nEnter adjecency matrix of the graph:");
 for(i=0;i<n;i++)
      for(j=0;j<n;j++)
scanf("%d",&a[i][j]);
     for(i=0;i<n;i++)
       visited[i]=0;
     printf("Enter the starting vertex\n");
     scanf("%d",&v);
    DFS(v);
    return 0;
}
```

Output:



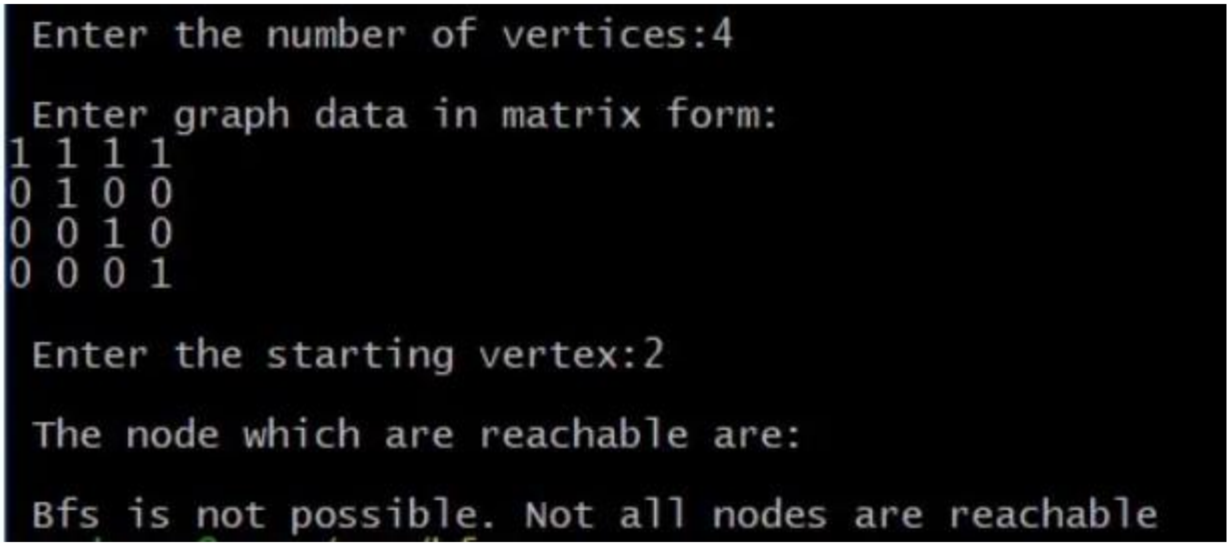
```
Enter number of vertices:8

Enter adjecency matrix of the graph:0 1 1 1 1 0 0 0
1 0 0 0 0 1 0 0
1 0 0 0 0 1 0 0
1 0 0 0 0 0 1 0
1 0 0 0 0 0 1 0
0 1 1 0 0 0 0 1
0 0 0 1 1 0 0 1
0 0 0 0 0 1 1 0

0
1
5
2
7
6
3
4
Process returned 8 (0x8)   execution time : 64.785 s
Press any key to continue.
```

5.Write a program implement the BFS algorithm for a graph.

```
#include<stdio.h>
int a[20][20],q[20],visited[20],n,f=-1,r=-1;
void bfs(int v)
{
    int k;
      for (k=0;k<n;k++)
      {
            if(visited[k] == 0 && a[v][k] == 1 )
            {
                  r=r+1;
                  q[r]=k;
                  visited[k]=1;
                  printf("%d  ",k);
            }
      }
      f=f+1;
      if(f<=r)
           bfs(q[f]);
}
int main()
{
   int v,i,j;
   printf("\n Enter the number of vertices:");
   scanf("%d",&n);
   printf("\n Enter graph data in matrix form:\n");
   for (i=0;i<n;i++)
    {
            for (j=0;j<n;j++)
            {
                  scanf("%d",&a[i][j]);
            }
    }
    for (i=0;i<n;i++)
   {
```

```
            visited[i]=0;
        }
        printf("\n Enter the starting vertex:");
        scanf("%d",&v);
        f=r=0;
        q[r]=v;
        printf("\n BFS traversal is:\n");
        visited[v]=1;
        printf("%d",v);
        bfs(v);
    }
```

Output:



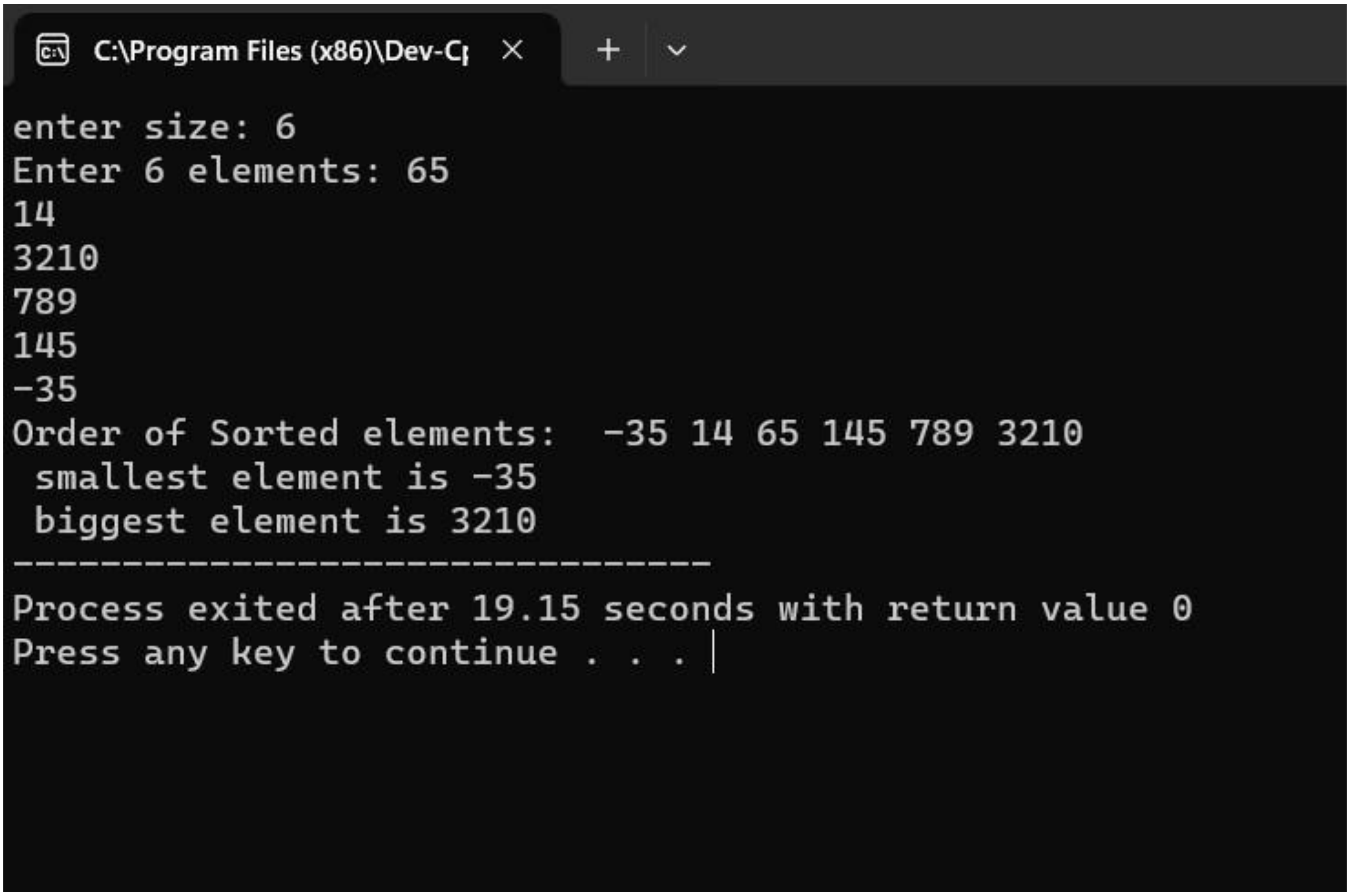
```
 Enter the number of vertices:4

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

 Enter the starting vertex:2

 The node which are reachable are:

 Bfs is not possible. Not all nodes are reachable
```

6. Write a program to find minimum and maximum value in an array using divide and conquer.

```
#include<stdio.h>
void quicksort(int a[25],int lb,int ub)
{
  int start=lb, end=ub, pivot=a[lb], temp;
  if(lb<ub)
  {
    pivot=lb;
    start=lb;
    end=ub;
    while(start<end)
     {
      while(a[start]<=a[pivot]&&start<ub)
        start++;
      while(a[end]>a[pivot])
        end--;
      if(start<end){
        temp=a[start];
        a[start]=a[end];
        a[end]=temp;
      }
    }
    temp=a[pivot];
    a[pivot]=a[end];
    a[end]=temp;
    quicksort(a,lb,end-1);
    quicksort(a,end+1,ub);
  }
}
int main()
{
  int i, n, a[25];
  printf("enter size: ");
  scanf("%d",&n);
```

```
   printf("Enter %d elements: ", n);
   for(i=0;i<n;i++)
      scanf("%d",&a[i]);
   quicksort(a,0,n-1);
   printf("Order of Sorted elements: ");
   for(i=0;i<n;i++)
      printf(" %d",a[i]);
        printf("\n smallest element is %d",a[0]);
    printf("\n biggest element is %d",a[n-1]);
       return 0;
}
```

Output:



```
enter size: 6
Enter 6 elements: 65
14
3210
789
145
-35
Order of Sorted elements:  -35 14 65 145 789 3210
 smallest element is -35
 biggest element is 3210
--------------------------------
Process exited after 19.15 seconds with return value 0
Press any key to continue . . .
```

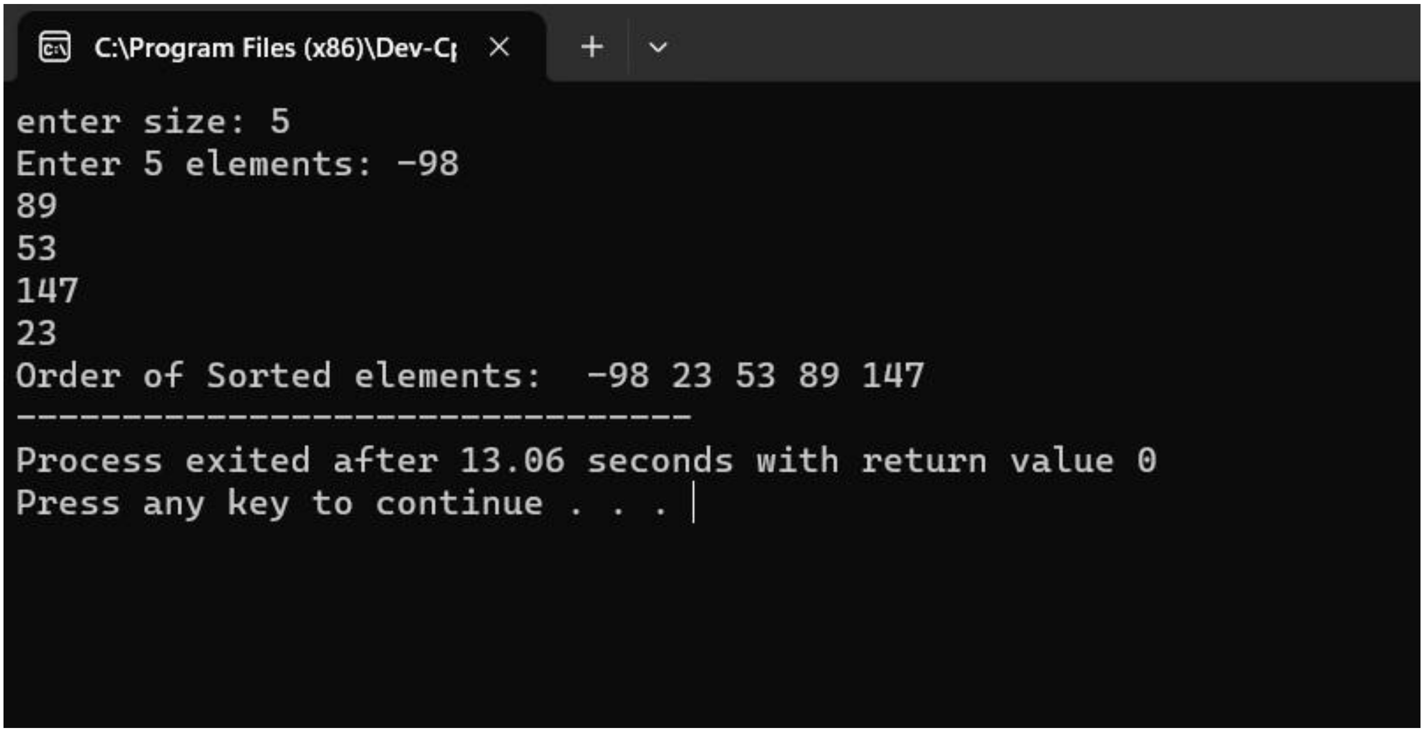
Part-B

1. Write a test program to implement Divide and Conquer Strategy for Quick sort algorithm.

```
#include<stdio.h>
void quicksort(int a[25],int lb,int ub){
  int start, end, pivot, temp;
  if(lb<ub){
    pivot=lb;
    start=lb;
    end=ub;
    while(start<end){
      while(a[start]<=a[pivot]&&start<ub)
        start++;
      while(a[end]>a[pivot])
        end--;
      if(start<end){
        temp=a[start];
        a[start]=a[end];
        a[end]=temp;
      }
    }
    temp=a[pivot];
    a[pivot]=a[end];
    a[end]=temp;
    quicksort(a,lb,end-1);
    quicksort(a,end+1,ub);
  }
}
int main(){
  int i, n, a[25];
  printf("enter size: ");
  scanf("%d",&n);
  printf("Enter %d elements: ", n);
  for(i=0;i<n;i++)
```

```
    scanf("%d",&a[i]);
  quicksort(a,0,n-1);
  printf("Order of Sorted elements: ");
  for(i=0;i<n;i++)
    printf(" %d",a[i]);
  return 0;
}
```

Output:



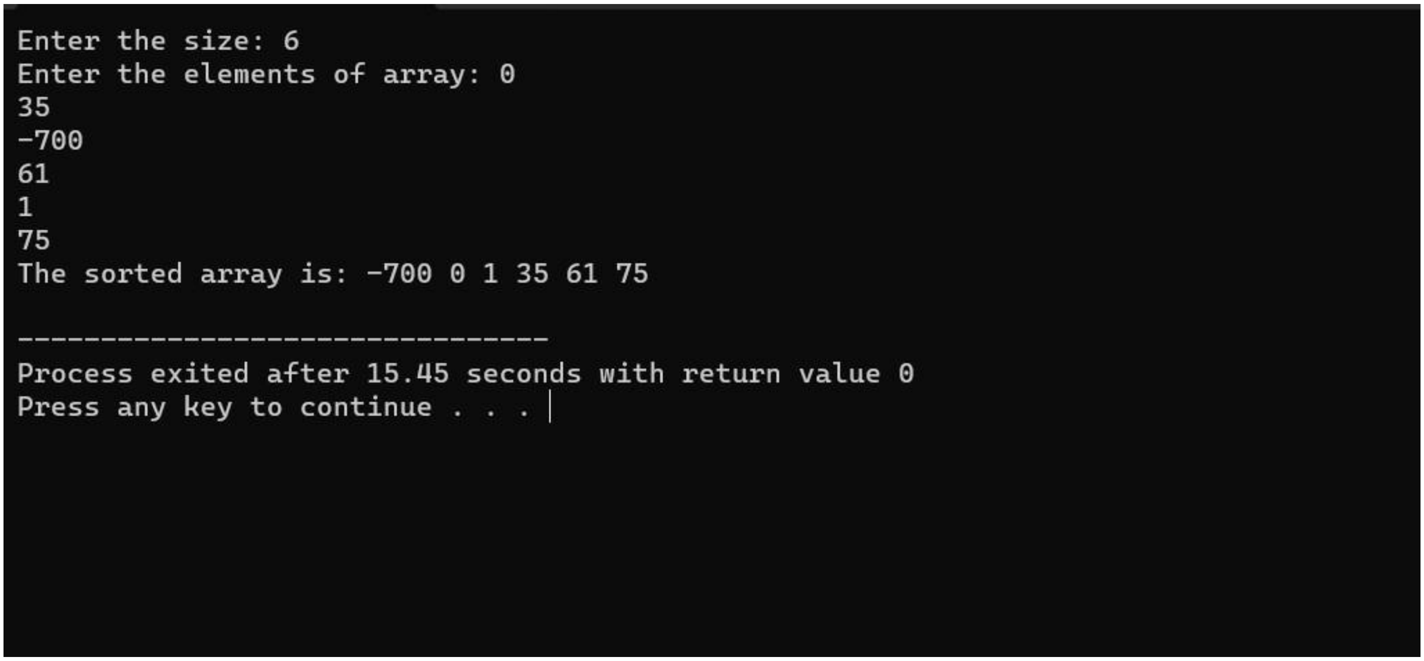
```
enter size: 5
Enter 5 elements: -98
89
53
147
23
Order of Sorted elements:  -98 23 53 89 147
--------------------------------
Process exited after 13.06 seconds with return value 0
Press any key to continue . . .
```

2. Write a program to implement Merge sort algorithm for sorting a list of integers in ascending order.

```
#include <stdio.h>
void Merge(int arr[], int lb, int mid, int ub)
{
    int i, j, k;
    int n1 = mid - lb + 1;
    int n2 = ub - mid;
     int Left_arr[n1], Right_arr[n2];
     for (i = 0; i < n1; i++)
        Left_arr[i] = arr[lb + i];
     for (j = 0; j < n2; j++)
        Right_arr[j] = arr[mid + 1 + j];
     i = 0;
    j = 0;
    k = lb;
    while (i < n1 && j < n2)
    {
        if (Left_arr[i] <= Right_arr[j])
        {
            arr[k] = Left_arr[i];
            i++;
        }
        else
        {
            arr[k] = Right_arr[j];
            j++;
        }
        k++;
    }
     while (i < n1)
    {
        arr[k] = Left_arr[i];
        i++;
        k++;
    }
```

```
     while (j < n2)
    {
        arr[k] = Right_arr[j];
        j++;
        k++;
    }
}
void divide(int arr[], int lb, int ub)
{
    if (lb < ub)
    {
         int mid = lb + (ub - lb) / 2;
        divide(arr, lb, mid);
        divide(arr, mid + 1, ub);
         Merge(arr, lb, mid, ub);
    }
}
 int main()
{
    int n;
    printf("Enter the size: ");
    scanf("%d", &n);
     int arr[n];
    printf("Enter the elements of array: ");
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
     divide(arr, 0, n - 1);
     printf("The sorted array is: ");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;}
```

Output:

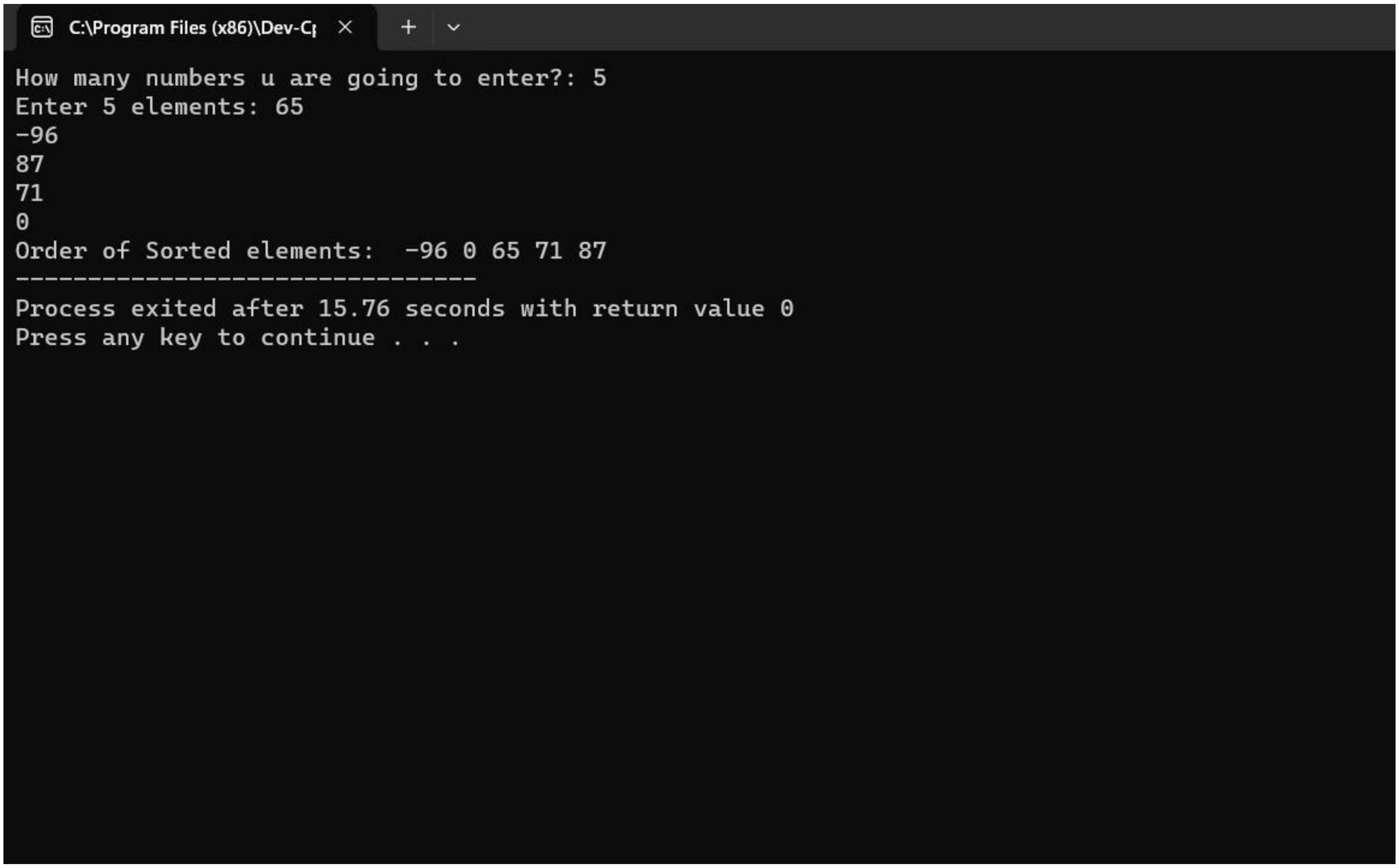


```
Enter the size: 6
Enter the elements of array: 0
35
-700
61
1
75
The sorted array is: -700 0 1 35 61 75

--------------------------------
Process exited after 15.45 seconds with return value 0
Press any key to continue . . .
```

3. Write a program to implement Insertion Sort.

```
#include<stdio.h>
int main(){
   int i, j, n, temp, a[25];
   printf("Enter the size ");
   scanf("%d",&n);
   printf("Enter %d elements: ", n);
   for(i=0;i<n;i++)
      scanf("%d",&a[i]);
   for(i=1;i<n;i++)
   {
      temp=a[i];
      j=i-1;
      while((temp<a[j])&&(j>=0))
       {
         a[j+1]=a[j];
         j--;
      }
      a[j+1]=temp;
   }
   printf("Order of Sorted elements: ");
   for(i=0;i<n;i++)
      printf(" %d",a[i]);
   return 0;
}
```

Output:

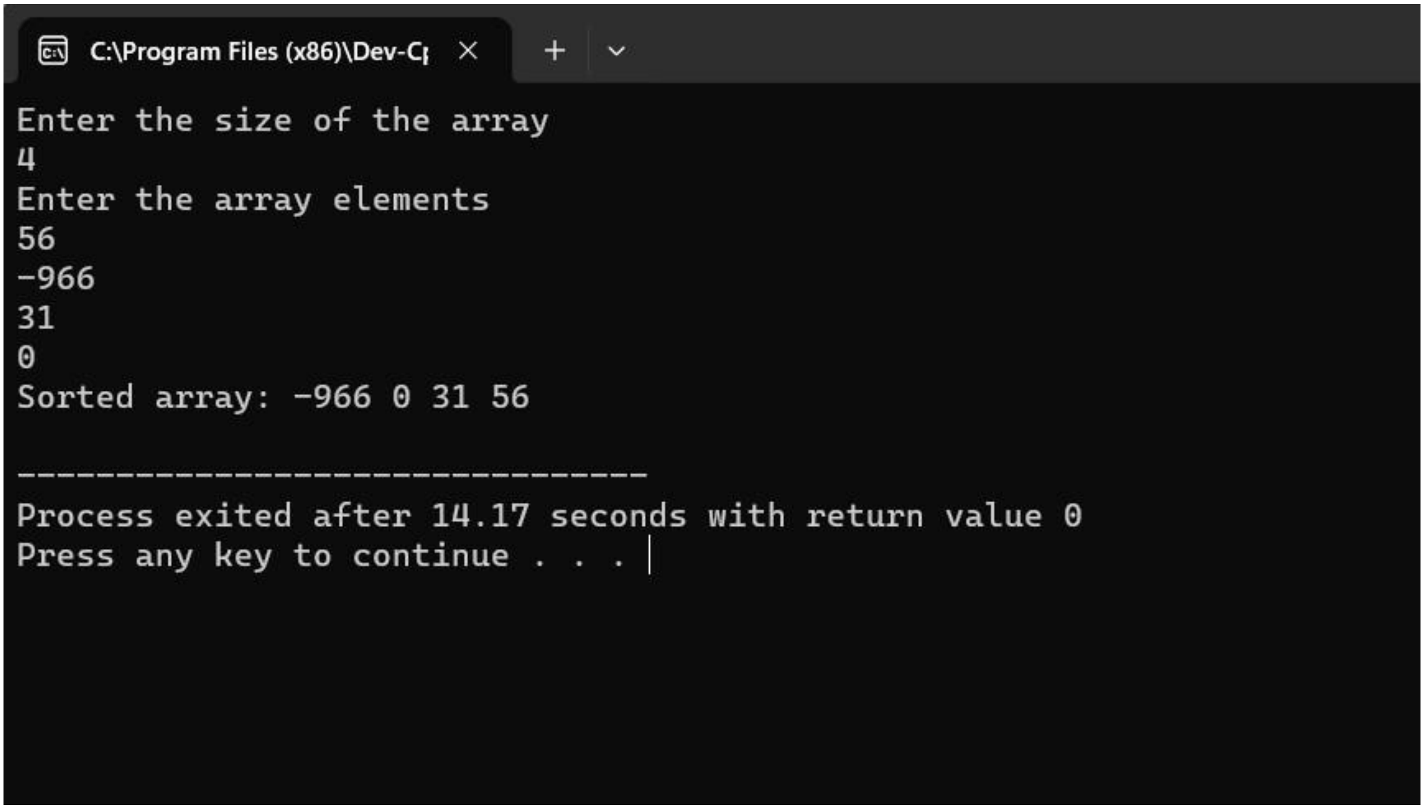


```
How many numbers u are going to enter?: 5
Enter 5 elements: 65
-96
87
71
0
Order of Sorted elements:  -96 0 65 71 87
--------------------------------
Process exited after 15.76 seconds with return value 0
Press any key to continue . . .
```

4. Write a program to implement Bubble Sort.

```
#include <stdio.h>
int main()
{
    int a[20],n,key,i,flag=0,index;
    printf("Enter the size of the array\n");
    scanf("%d",&n);
    printf("Enter the array elements\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
     for (int i = 0; i < n- 1; i++)
     {
        for (int j = 0; j < n- i - 1; j++)
                {
                          if (a[j] > a[j + 1]) {
                                int temp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = temp;
            }
        }
                }
     printf("Sorted array: ");
         for (int i = 0; i <n; i++)
                {
             printf("%d ", a[i]);
         }
         printf("\n");
}
```

Output:



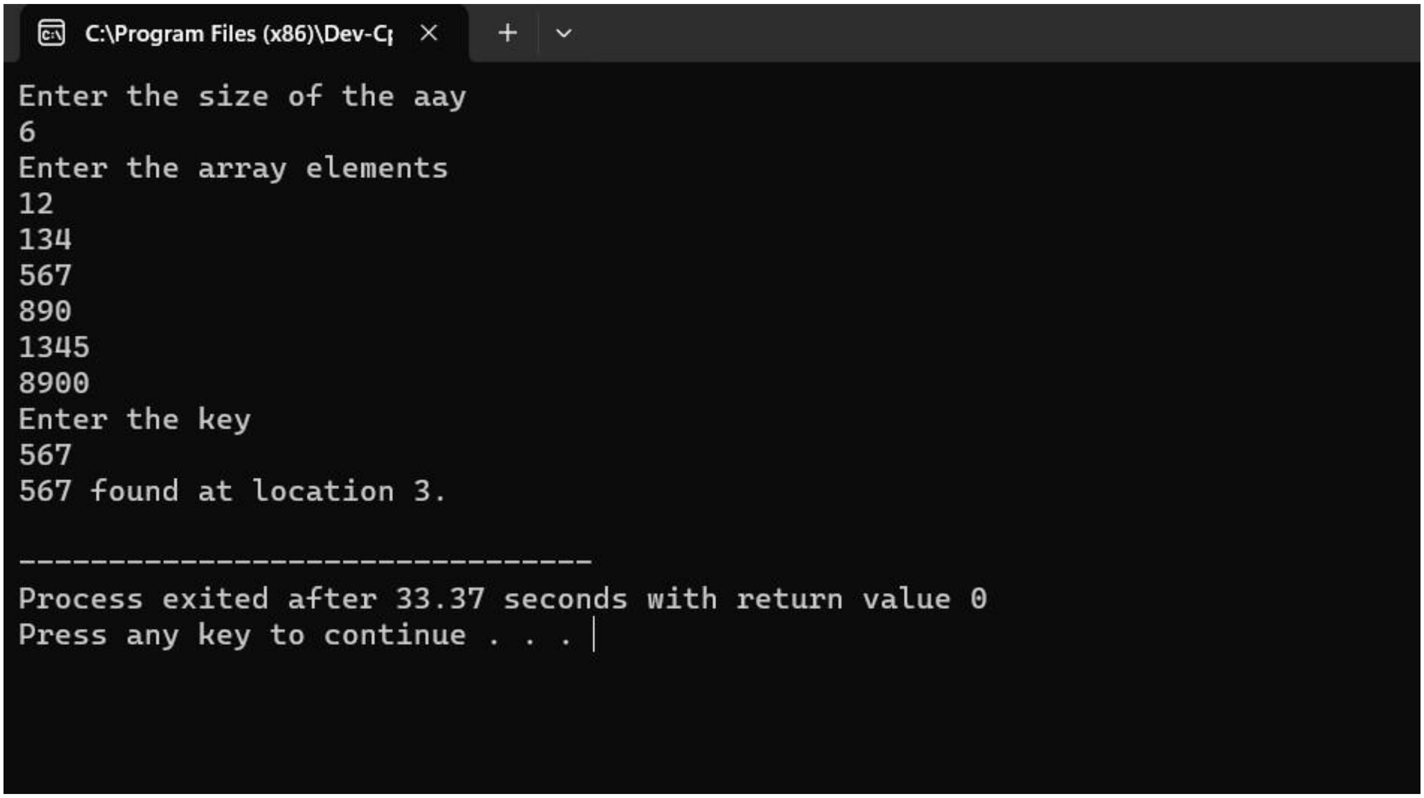
```
Enter the size of the array
4
Enter the array elements
56
-966
31
0
Sorted array: -966 0 31 56

--------------------------------
Process exited after 14.17 seconds with return value 0
Press any key to continue . . .
```

5.Write a program to implement Dynamic Programming algorithm for the Optimal Binary Search Tree Problem.

```
#include <stdio.h>
int main()
 {
   int a[20],n,first,last,flag=0, middle,i,index,key;
   printf("Enter the size of the aay\n");
   scanf("%d",&n);
   printf("Enter the array elements\n");
   for(i=0;i<n;i++)
   {
      scanf("%d",&a[i]);
   }
   printf("Enter the key\n");
   scanf("%d",&key);
   first = 0;
     last = n - 1;
     middle = (first+last)/2;

     for (;first <= last;)
      {
       if (a[middle] < key)
         first = middle + 1;
       else
            if (a[middle] == key)
            {
         printf("%d found at location %d.\n",key, middle+1);
         flag=1;
         break;
       }
       else
       {
           last = middle - 1;
           }
```

```
            middle = (first + last)/2;
          }
          if(flag!=1)
          {
                printf("Key not found\n");
          }
      return 0;
   }
```

Output:



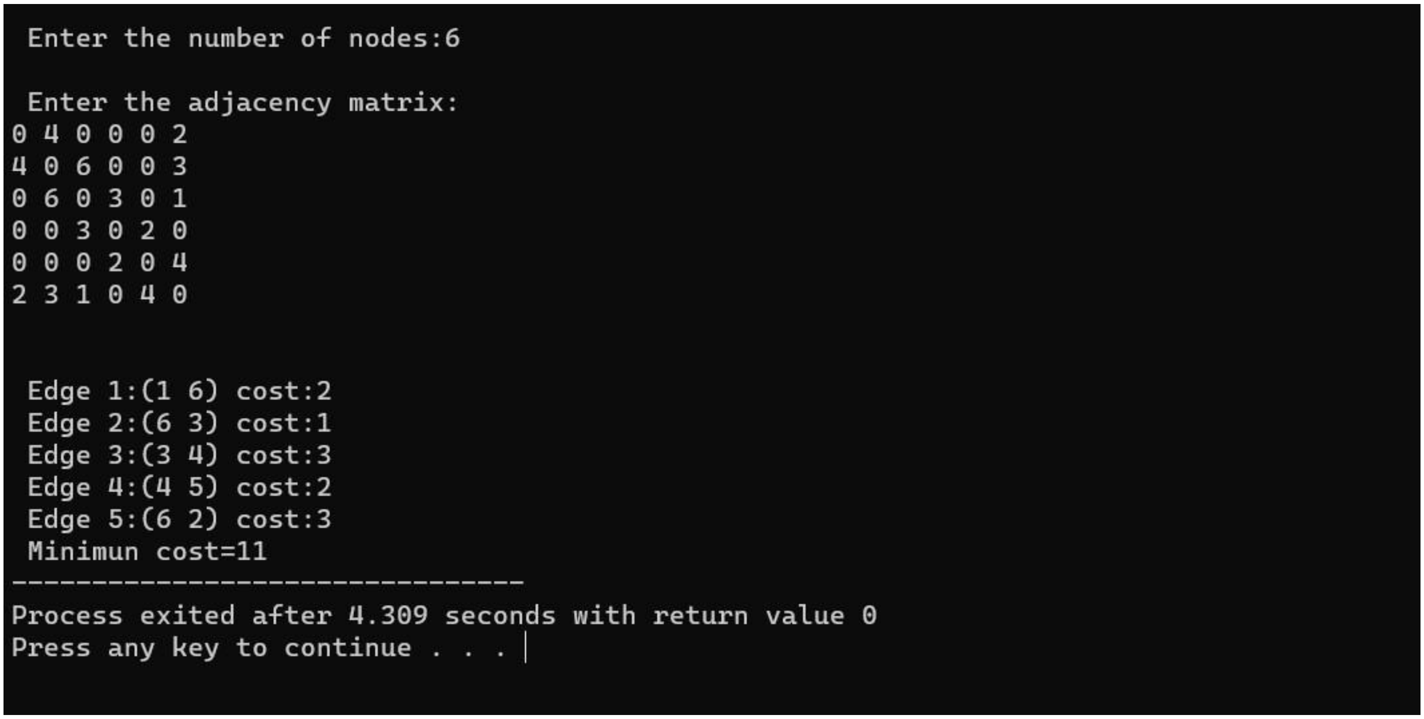
```
Enter the size of the aay
6
Enter the array elements
12
134
567
890
1345
8900
Enter the key
567
567 found at location 3.

--------------------------------
Process exited after 33.37 seconds with return value 0
Press any key to continue . . .
```

6. Write a program that implement Prim's algorithm to generate minimum cost spanning Tree.

```
#include<stdio.h>
int a,b,u,v,n,i,j,ne=1;
int visited[10]= {
    0
}
,min,mincost=0,cost[10][10];
int main()
{
    printf("\n Enter the number of nodes:");
    scanf("%d",&n);
    printf("\n Enter the adjacency matrix:\n");
    for (i=1;i<=n;i++)
      for (j=1;j<=n;j++) {
            scanf("%d",&cost[i][j]);
            if(cost[i][j]==0)
                cost[i][j]=999;
    }
    visited[1]=1;
    printf("\n");
    while(ne<n) {
            for (i=1,min=999;i<=n;i++)
               for (j=1;j<=n;j++)
                if(cost[i][j]<min)
                 if(visited[i]!=0) {
                      min=cost[i][j];
                      a=u=i;
                      b=v=j;
            }
            if(visited[u]==0 || visited[v]==0) {
                      printf("\n Edge %d:(%d %d) cost:%d",ne++,a,b,min);
                      mincost+=min;
                      visited[b]=1;
            }
```

```
                cost[a][b]=cost[b][a]=999;
        }
        printf("\n Minimun cost=%d",mincost);
        return 0;
    }
```

Output:



```
 Enter the number of nodes:6

 Enter the adjacency matrix:
0 4 0 0 0 2
4 0 6 0 0 3
0 6 0 3 0 1
0 0 3 0 2 0
0 0 0 2 0 4
2 3 1 0 4 0


 Edge 1:(1 6) cost:2
 Edge 2:(6 3) cost:1
 Edge 3:(3 4) cost:3
 Edge 4:(4 5) cost:2
 Edge 5:(6 2) cost:3
 Minimun cost=11
--------------------------------
Process exited after 4.309 seconds with return value 0
Press any key to continue . . .
```