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

```
#include<stdio.h>
int main()
{
   int a[100],i,n,j,temp,min;

   printf("\nEnter a number : ");
   scanf("%d",&n);

   for(i=0; i<n; i++)
   {
       printf("\nEnter[%d] an element : ",i);
       scanf("%d",&a[i]);
   }

   for(i=0; i<n-1; i++)
   {
      min = i;
      for(j=i+1; j<n; j++)
      {
         if(a[j] < a[min])
         {
            min = j;
         }
      }
         temp = a[i];
         a[i] = a[min];
         a[min] = temp;
   }

      for(i=0; i<n; i++){
      printf("\n [%d] index element is %d",i,a[i]);
   }
}
```

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

```
#include<stdio.h>
int a[10][10],visited[10],n; //n is no of vertices and graph is sorted in array
G[10][10];
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);
      //read the adjecency matrix
     printf("\nEnter adjecency matrix of the graph:");

   for(i=0;i<n;i++)
   {
     for(j=0;j<n;j++)
     {
         scanf("%d",&a[i][j]);
      }
   }
   //visited is initialized to zero
   for(i=0;i<n;i++)
   {
       visited[i]=0;
   }

   printf("Enter the starting vertex\n");
   scanf("%d",&v);
   DFS(v);
   return 0;
}
```

3.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;
}
```

4. Write C program that accepts the vertices and edges for a graph and stores it as an adjacency matrix.

```
#include<stdio.h>
int main()
{
    int i,j,n,a[20][20];
    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]);
      }
   }

   printf("matrix is \n");
   for(i=0;i<n;i++)
   {
      for(j=0;j<n;j++)
      {
         printf("%d",a[i][j]);
      }
      printf("\n");
   }

 return 0;
}
```

5.Implement function to print In-Degree, Out-Degree and to display that adjacency matrix.

```
#include<stdio.h>
int main()
 {
   int i, j, n, a[20][20], sum1, sum2;

   printf("Enter number of vertices:");
   scanf("%d", &n);

   // Read the adjacency matrix
   printf("\nEnter adjacency matrix of the graph:");
   for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
          scanf("%d", &a[i][j]);
        }
   }


   // Calculate out-degree
   printf("\nOut-degree:\n");
   for (i = 0; i < n; i++)
    {
         sum1 = 0;
         for (j = 0; j < n; j++)
         {
             sum1 += a[i][j];
         }
      printf("Vertex %d: %d\n", i + 1, sum1);
   }

   // Calculate in-degree
   printf("\nIn-degree:\n");
   for (i = 0; i < n; i++)
   {
         sum2 = 0;
         for (j = 0; j < n; j++)
         {
             sum2 += a[j][i];
          }
      printf("Vertex %d: %d\n", i + 1, sum2);
   }
   return 0;
}
```

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

```
#include<stdio.h>
void quicsort(int a[20],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;
      quicsort(a,lb,end-1);
      quicsort(a,end+1,ub);
   }
}

int main()
{
   int a[20],i,n;

   printf("\nEnter a number : ");
   scanf("%d",&n);

   for(i=0; i<n; i++)
   {
      printf("\nEnter an elment : ");
      scanf("%d",&a[i]);
   }
   quicsort(a,0,n-1);
```

```
    printf("\nSorted elements : ");

    for(i=0; i<n; i++)
    {
       printf("%d ",a[i]);
    }

    printf("\nsmallest elements : %d ",a[0]);
    printf("\nbiggest elements : %d ",a[n-1]);
    return 0;
}
```

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

```
#include<stdio.h>
void quicsort(int a[20],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;
      quicsort(a,lb,end-1);
      quicsort(a,end+1,ub);
   }
}
int main()
{
   int a[20],i,n;

   printf("\nEnter a number : ");
   scanf("%d",&n);

   for(i=0; i<n; i++)
   {
       printf("\nEnter an elment : ");
       scanf("%d",&a[i]);
   }

   quicsort(a,0,n-1);
   printf("\nSorted elements : ");
```

```
   for(i=0; i<n; i++)
   {
        printf("%d ",a[i]);
   }
   return 0;
}
```

8.Write a program that implements 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:");

   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;
}
```

PhonePe

ACCEPTED HERE

Scan & Pay Using PhonePe App

Sachikphonep

दस रुपये भेजने से आप गरीब नहीं हो जायेंगे