First Semester MCA (2020 scheme) Practical Examination
June 2021.

20MCA 135 Data Structures Lab

Date : 30/06/2021. Time : 9.30 AM to 12.30 PM.

Name : Akhila P M. Reg No : ICE20MCA2006.

Batch A.

Sorting of an Integer Array.

Program:

```
#include <stdio.h>
#include <conio.h>
Void main()
{
int i,j,x,n,a[30];
clrscr();
Printf("enter the total number of elements:\n");
scanf("%d",&n);
Printf("enter the elements\n");
for(i=0;i<n;i++)
{
```

```
{
scanf("%d",&a[i]);
}
for(i=0;i<n;i++)
{
for(j=i+1;j<n;j++)
{
if(a[i]>a[j])
{
x=a[i];
a[i]=a[j];
a[j]=x;
}
}
}
printf("The elements in assending order \n");
for(i=0;i<n;i++)
{
printf("%d \n",a[i]);
}
getch();
}.
```

output.

enter the total no: of elements.

5.

Enter the elements.

the elements in ascending order

0
2
4
7
9.

2. Implementing Prims algorithm.

```
#include<stdio.h>
#include<conio.h>
int a,b,u,v,n,i,j,ne=1;
int visited[10]={0},min,mincost=0,cost[10][10];
void main()
{
clrscr();
Printf("\n Enter the no: 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 Minimum cost %d", mincost);
getch();
}
```

Output:

```
Enter the no: of nodes : 6.
Enter the adjacency Matrix:

0 3 1 6 0 0
3 0 5 0 3 0
1 5 0 5 6 1
6 0 5 0 0 2
0 3 6 0 0 6
0 0 4 2 6 0.

Edge 1 : (1 3) cost : 1
Edge 2 : (3 6) cost : 1
Edge 3 : (6 4) Cost : 2
Edge 4 : (1 2) Cost : 3
Edge 5 : (2 5) Cost : 3
Minimum Cost 10.
```

```
printf("\n Minimum cost %d", mincost);
getch();
}
```

Output:

```
Enter the no: of nodes : 6.
Enter the adjacency Matrix:
0 3 1 6 0 0
3 0 5 0 3 0
1 5 0 5 6 1
6 0 5 0 0 2
0 3 6 0 0 6
0 0 4 2 6 0.

Edge 1 : (1 3) cost : 1
Edge 2 : (3 6) cost : 1
Edge 3 : (6 4) Cost : 2.
Edge 4 : (1 2) Cost : 3
```