FIRST SEMESTER MCA PRATICAL EXAMINATION - JUN-JUL -2021

20MCA135 DATA SIRUCTURE LAB

Batch - 1

Najmal Fawaz
ICE20MCA-2030

Date : 30 June 2021

Time : 1.00PM - 4.00Pm

Q:1) Merging of two Sorted Array.

Q:2) Implement prims Algorithm.

Q:1) Aim : Merging of two Sorted Array.

Pgm:

```
#Include <Stdio.h>
#Include <conio.h>
 void main ():
{
Int array1 [50], array2 [50], array3 [100], m,n,i,j,k=0;
clrscr();
printf ("\n Enter size of array :");
Scanf("%d", &m);
printf ("\n Enter Sorted elements of array1 :\n");
for (i=0; i<m; i++)
{
Scanf ("%d", &array1 [i]);
}
printf ("\n Enter size of array2: ");
Scanf ("%d", &n);
printf ("\n Enter Sorted elements of array2: \n");
 for (i=0; i<n; i++)
{
```

```
Sconf ("%d", &array2[i]);
}
i=0;
j=0;
while (i<m && j<n)
{
if (array1[i]<array2[j])
{
array3[k] = array1[i];
i++;
}
else
{
array3[k] = array2[j];
j++;
}
k++;
}
if (i>=m)
{
while (j<n)
{
array3[k]=array2[j];
j++;
k++;
}
}
if (j>=n)
{
```

```
while (i<m)
{
array3[k] = array1[i];
i++;
k++;
}
}
printf("\n After merging: \n");
for(i=0; i<m+n; i++)
{
printf("\n %d", array3[i]);
}
getch();
}
```

Output:

Enter the size of array1 : 3

Enter Sorted elements of array1:

2 3 4

Enter the Size of array2 : 3

Enter Sorted elements of array2:

5 6 7

After Merging:

2
3
4
5
6
7

9) Aim. Implement of prims Algorithm

Program:

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

Output :

Enter the number of nodes : 6

Enter the adjacency 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

Edge 1 : (1 3) Cost : 1

Edge 2 : (1 2) Cost : 3

Edge 3 : (2 5) Cost : 3

Edge 4 : (3 6) Cost : 4

Edge 5 : (6 4) Cost : 2

Minimum Cost : 13