First Semester MCA (2020 scheme) Practical Examination

June 2021

20MCA135 DataStructure Lab

Date : 30/06/2021 Time : 1:00 PM to 4:00 PM

Name : Ramsina yoosuf RegNo : ICE20MCA2032

Batch 2

1) Merging of 2 sorted arrays

2) Implement Prims Algorithm.

Answers

1) Merging of 2 Sorted arrays.

```
#include<stdio.h>
#include<conio.h>
void main()
{
 ~~int arr[50]~~
 int array1[50],array2[50],array3[100],i,j,k=0,m,n;
 clrscr();
 printf("enter the size of first array:");
 scanf("%d",&m);
 printf("enter the sorted elements of first array\n");
 for(i=0;i<m;i++)
 {
  scanf("%d",&array1[i];
 }
 printf("enter the size of second array:");
```

```
Scanf("%d", &n);
Printf ("enter the sorted elements of second array");
for(i=0;i<n;i++)
{
Scanf("%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]=array2[i];
  i++;
  k++;
 }
 }
  printf("\n After Merging:\n");
  for(i=0;i<m+n;i++)
  {
  printf("\n %d", array3[i]);
  }
  getch();
 }
```

<u>Output</u>

```
enter the size of first array:3
enter the sorted elements of first array
 1 2 3
a
enter the size of second array:3
enter the sorted elements of second array
4 5 6
```

After merging:

```
1
2
3
4
5
6
```

2) Implement prims Algorithm

```
#include<stdio.h>
#include<conio.h>

int a,b,u,v,n,i,j,ne=1;
void 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<~~ 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 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
```