```
#include<iostream>
using namespace std;
int parent_cost,r,no_of_nodes;
int c[10][10],temp[10][10],visited[10],path[10];
void disp(int x[10][10]);
void copy(int x[10][10],int y[10][10]);
void cal_cost(int set[],int n);
int check_reduced();
int flag=1;
int main()
{
	int i,j,set[10];
	cout<<"\n Enter the no. of nodes in the graph:";
	cin>>no_of_nodes;
	for(i=1;i<=no_of_nodes;i++)
	{
		for(j=1;j<=no_of_nodes;j++)
		{
			if(i!=j)
			{
				cout<<"\n Enter the cost of edge between";
				cout<<i<<"-"<<j<<":";
				cin>>c[i][j];
			}
			else
				c[i][j]=99;
			}
		}
```

```
        visited[1]=1;
        path[r]=1;
        r++;
        disp(c);
        copy(temp,c);
        int root_cost=check_reduced();
        parent_cost=root_cost;
        copy(c,temp);
        cout<<"\n Reduced cost matrix:\n";
        disp(c);
        int p=0;
        for(i=1;i<=no_of_nodes;i++)
        {
                if(visited[i]!=1)
                {
                        set[p]=i;
                        p++;
                }
        }
        cal_cost(set,p);
        cout<<"\n Minimum cost:"<<parent_cost;
        cout<<"\n \n Resultant path:";for(i=0;i<r;i++)
        cout<<path[i]<<"->";
        cout<<"1";
        return 0;
}
void disp(int x[10][10])
{
```

```
	for(int i=1;i<=no_of_nodes;i++)
	{
		cout<<"\t";
		for(int j=1;j<=no_of_nodes;j++)
		cout<<x[i][j]<<"\t";
		cout<<"\n";
	}
}
void copy(int x[10][10],int y[10][10])
{
	for(int i=1;i<=no_of_nodes;i++)
	{
		for(int j=1;j<=no_of_nodes;j++)
		{
			x[i][j]=y[i][j];
		}
	}
}
int find_min(int g[10],int n,int q[10])
{
	int min=g[0];
	int node=q[0];
	for(int i=1;i<n;i++)
	{
		if(min>g[i])
		{
			min=g[i];
			node=q[i];
```

```
                }
        }
        path[r]=node;
        r++;
        visited[node]=1;
        return(min);
}
int check_reduced()
{
        int min,sum=0;
        for(int i=1;i<=no_of_nodes;i++)
        {
                min=999;
                for(int j=1;j<=no_of_nodes;j++)
                {
                        if(temp[i][j]<min)
                                min=temp[i][j];
                }
                if(min!=0 && min!=99)
                {
                        for(int j=1;j<=no_of_nodes;j++)
                        {
                        if(i!=j && temp[i][j]!=99)
                        temp[i][j]=temp[i][j]-min;
                        }
                        sum=sum+min;
                }
        }
```

```
	for(int j=1;j<=no_of_nodes;j++)
	{
		min=999;
		for(int i=1;i<=no_of_nodes;i++)
		{
			if(temp[i][j]<min)
			min=temp[i][j];
		}
		if(min!=0 && min!=99)
		{
			for(int i=1;i<=no_of_nodes;i++)
			{
				if(j!=i && temp[i][j]!=99)
				temp[i][j]=temp[i][j]-min;
			}
		sum=sum+min;
		}
	}
	cout<<"\n\n";
	disp(temp);
	cout<<"\nsum:"<<sum;
	return(sum);
}
void cal_cost(int set[],int n)
{
	int g[10],q[10];
	for(int i=0;i<n;i++)
	{
```

```
        copy(temp,c);
        for(int j=0;j<r;j++)
        {
                for(int k=1;k<=no_of_nodes;k++)
                {
                        temp[path[j]][k]=99;
                        //temp[k][i]=99;
                        if(j!=0)
                        temp[k][path[j]]=99;
                }
        }
        for(int k=1;k<=no_of_nodes;k++)
        temp[k][set[i]]=99;
        temp[set[i]][1]=99;//column
        int ans=check_reduced();
        g[i]=parent_cost+ans+c[path[r-1]][set[i]];
        cout<<"cost"<<g[i];
        q[i]=set[i];
}
parent_cost=find_min(g,n,q);
int p=0;
for(int i=1;i<=no_of_nodes;i++)
{
        if(visited[i]!=1)
        {
        set[p]=i;
        p++;
        }
```

```
        }
        if(p!=0)
        cal_cost(set,p);
}


/*

pc:~$ g++ Assignment8.cpp
pc:~$ ./a.out

 Enter the no. of nodes in the graph:4

 Enter the cost of edge between1-2:3

 Enter the cost of edge between1-3:2

 Enter the cost of edge between1-4:5

 Enter the cost of edge between2-1:7

 Enter the cost of edge between2-3:2

 Enter the cost of edge between2-4:3

 Enter the cost of edge between3-1:2

 Enter the cost of edge between3-2:4
```

Enter the cost of edge between3-4:6

Enter the cost of edge between4-1:8

Enter the cost of edge between4-2:5

Enter the cost of edge between4-3:2

99	3	2	5
7	99	2	3
2	4	99	6
8	5	2	99

99	0	0	2
5	99	0	0
0	1	99	3
6	2	0	99

sum:10

Reduced cost matrix:

99	0	0	2
5	99	0	0
0	1	99	3
6	2	0	99

99	99	99	99

99	99	0	0
0	99	99	3
6	99	0	99

sum:0cost10

99	99	99	99
1	99	99	0
99	0	99	2
0	0	99	99

sum:7cost17

99	99	99	99
5	99	0	99
0	0	99	99
99	1	0	99

sum:1cost13

99	99	99	99
99	99	99	99
99	99	99	0
0	99	99	99

sum:9cost19

99	99	99	99

```
        99      99      99      99
        0       99      99      99
        99      99      0       99

sum:0cost10

        99      99      99      99
        99      99      99      99
        99      99      99      99
        99      99      99      99

sum:0cost10
 Minimum cost:10

 Resultant path:1->2->4->3->1

 */
```