#include<iostream>

using namespace std;

class kruskal

{

struct node

{

int a,b;

int weight;

};

struct node store[100],path[100];

int parent[100];

int pn,nv,ne;

public:

kruskal()

{

pn=0;

for(int x=1;x<100;x++)

{

parent[x]=x;

}

}

void create()

{

int i;

cout<<"enter no of vertices";

cin>>nv;

cout<<"enter no of edges";

cin>>ne;

cout<<"Enter the strting vetice ,ending vetice and weight"<<endl;

for(i=0;i<ne;i++)

{

cout<<"Enter"<<i+1<<"set"<<endl;

cin>>store[i].a>>store[i].b>>store[i].weight;

}

}

void sort()

{

int i,j;

struct node temp;

for(i=0;i<ne;i++)

{

for(j=0;j<ne-i-1;j++)

{

if(store[j].weight>store[j+1].weight)

{

temp=store[j];

store[j]=store[j+1];

store[j+1]=temp;

}

}

}

}

void kalgo()

{

sort();

pn=0;int x=0;

while(pn<nv-1)

{

int a=find(store[x].a);

int b=find(store[x].b);

if(a!=b)

{

path[pn++]=store[x];

unionc(a,b);

}

x++;

}

}

int find(int x)

{

if(parent[x]!=x)

{

parent[x]=find(parent[x]);

}

else

return parent[x];

}

void unionc(int i,int j)

{

parent[j]=i;

}

void display()

{

int i;

cout<<"Starting Vertices\tEnding Vertices\tweight"<<endl;

for(i=0;i<pn;i++)

{

cout<<path[i].a<<"\t\t"<<path[i].b<<"\t\t"<<path[i].weight<<"\n";

}

}

int cost()

{

int i,sum=0;

for(i=0;i<pn;i++)

{

sum=sum+path[i].weight;

}

return sum;

}

};

int main()

{

kruskal K;

int ch,c;

do{

cout<<"1.Create the Graph"<<endl;

cout<<"2.Calculate miniming spanning tree"<<endl;

cout<<"3.Display the path"<<endl;

cout<<"4.Cost of the minimum spaaning tree"<<endl;

cout<<"Enter the choice"<<endl;

cin>>ch;

switch(ch)

{

case 1:

K.create();

break;

case 2:

K.kalgo();

break;

case 3:

K.display();

break;

case 4:

c=K.cost();

cout<<"The Cost of the minimum spaaning tree is-"<<c<<endl;

break;

}

}while(ch!=5);

return 1;

}

Output:

