ASSIGNMENT 4

Aim:-

. For a weighted graph G, find the minimum spanning tree using Prims algorithm

Objective:-

To find the minimum spanning tree using Prims algorithm

Theory:-

Prim's algorithm is a minimum spanning tree algorithm that takes a graph as input and finds the subset of the edges of that graph which

- form a tree that includes every vertex
- has the minimum sum of weights among all the trees that can be formed from the graph

Algorithm:-

t falls under a class of algorithms called greedy algorithms which find the local optimum in the hopes of finding a global optimum.

We start from one vertex and keep adding edges with the lowest weight until we we reach our goal.

The steps for implementing Prim's algorithm are as follows:

1. Initialize the minimum spanning tree with a vertex chosen at random.
2. Find all the edges that connect the tree to new vertices, find the minimum and add it to the tree
3. Keep repeating step 2 until we get a minimum spanning tree.

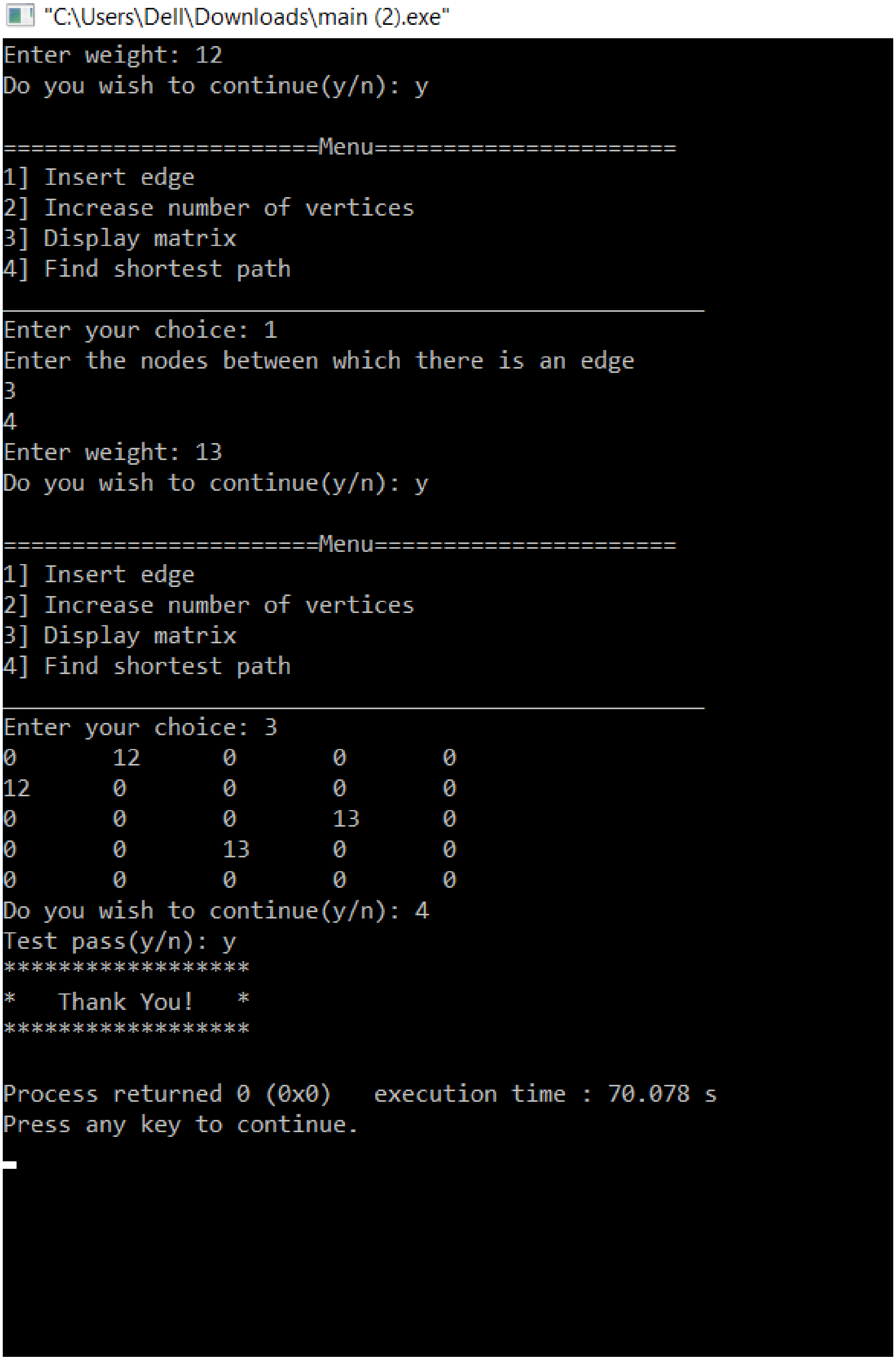
Code:-

```
        #include <iostream>
using namespace std;
class graph
{
    int a[100][100];
    int v;
public:
    void insert_edge(int n1,int n2,int wt)
    {
        if(n1-1>=v||n2-1>=v)
            cout<<"Vertex request out of range\n";
        else
        {
            a[n1-1][n2-1]=wt;
            a[n2-1][n1-1]=wt;
        }
    }
    void display()
    {
        for(int i=0;i<v;i++)
        {
            for(int j=0;j<v;j++)
            {
                cout<<a[i][j]<<"\t";
            }
            cout<<endl;
        }
    }
    void update_v(int n)
    {
        v=n;
    }
    void prims(int src)
    {
        int sp[v],dist[v],visited[v],parent[v],c=0;
        for(int i=0;i<v;i++)
        {
            visited[i]=0;
            dist[i]=9999;
        }
        dist[src-1]=0;
        parent[src-1]=-1;
        for(int i=0;i<v;i++)
        {
            int min=9999,min_ind;
            for(int j=0;j<v;j++)
            {
                if(!visited[j] && dist[j]<min )
                {

                    min=dist[j];
                    min_ind=j;
                }
            }
            int U=min_ind;
            visited[U]=1;
            sp[c]=U;
            c++;
            for(int V=0;V<v;V++)
            {
                if(!visited[V] && a[U][V] && a[U][V]<dist[V] &&
dist[U]!=9999)
```

```
                  {
                    parent[V]=U;
                    dist[V]=a[U][V];
                  }
            }
        }
        for(int i=0;i<c;i++)
        {
            cout<<sp[i]+1<<" link from "<<parent[i]+1<<endl;
        }
        cout<<endl;
    }
};
int main()
{
    char r;
    do
    {
        graph g;
        char op;
        int v;
        cout<<"Enter number of vertices: ";
        cin>>v;
        g.update_v(v);
        do
        {
            int c;
            cout<<"\n=======================Menu======================\n";
            cout<<"1] Insert edge\n2] Increase number of vertices\n3]
Display matrix\n4] Find shortest path\n";
            cout<<"___________________________________________________\n";
            cout<<"Enter your choice: ";
            cin>>c;
            switch(c)
            {
                case 1: {
                            int n1,n2,wt;
                            cout<<"Enter the nodes between which there is
an edge\n";
                            cin>>n1>>n2;
                            cout<<"Enter weight: ";
                            cin>>wt;
                            g.insert_edge(n1,n2,wt);
                        }
                        break;
                case 2: {
                            int n;
                            cout<<"Enter the number by which you wish to
increase the vertices: ";
                            cin>>n;
                            v+=n;
                            g.update_v(v);
                        }
                        break;
                case 3: {
                            g.display();
                        }
                        break;
                case 4: {
                            int src,dst;
                            cout<<"Source: ";
                            cin>>src;
                            g.prims(src);
                        }
                        break;
                default:cout<<"Error 404.....page not found\n";
            }
            cout<<"Do you wish to continue(y/n): ";
            cin>>op;
        }while(op=='y' || op=='Y');
        cout<<"Test pass(y/n): ";
        cin>>r;
    }while(r=='n' || r=='N');
    cout<<"******************\n";
    cout<<"*   Thank You!   *\n";
    cout<<"******************\n";
    return 0;
}
```

Output Screenshot:-



"C:\Users\Dell\Downloads\main (2).exe"

```
Enter weight: 12
Do you wish to continue(y/n): y

=======================Menu======================
1] Insert edge
2] Increase number of vertices
3] Display matrix
4] Find shortest path
___________________________________________________
Enter your choice: 1
Enter the nodes between which there is an edge
3
4
Enter weight: 13
Do you wish to continue(y/n): y

=======================Menu======================
1] Insert edge
2] Increase number of vertices
3] Display matrix
4] Find shortest path
___________________________________________________
Enter your choice: 3
0       12      0       0       0
12      0       0       0       0
0       0       0       13      0
0       0       13      0       0
0       0       0       0       0
Do you wish to continue(y/n): 4
Test pass(y/n): y
******************
*   Thank You!   *
******************

Process returned 0 (0x0)   execution time : 70.078 s
Press any key to continue.
```

Conclusion:-

We Have Successfully Implemented Prims Algorithm.