```cpp
/// Prim's Algorithm(MST):
#define mx 100
vector<int>adj[mx]; int cost[mx][mx]; queue<int>Q; bool vis[mx];
struct edge{
    int u,v,w;
    edge(int x, int y, int c){
        u=x;v=y;w=c;
    }
    bool operator < (const edge& p) const{
        return p.w < w;
    }
};

int primsMST(int node, int src){
    priority_queue<edge>pq;
    int Mincost=0;

    for(int i=0; i<adj[src].size(); i++){
        int v = adj[src][i];
        pq.push(edge(src,v,cost[src][v]));
    }

    for(int j=1; j<=node-1; j++){
        label:
        edge d(0,0,0);
        d = pq.top();
        pq.pop();
        if(vis[d.u]==true&&vis[d.v]==true){ //create cycle
            pq.pop();
            goto label;
        }
        vis[d.u]=true; vis[d.v]=true;
        Q.push(d.u); Q.push(d.v);
        Mincost+=d.w;
        int u = d.v;
        for(int i=0; i<adj[u].size(); i++){
            int v = adj[u][i];
            if(vis[v]==false){
                pq.push(edge(u,v,cost[u][v]));
            }
        }
    }
    return Mincost;
}
int main(){
    int node,edge,u,v,w,mincst;
    cin >> node >> edge;

    for(int i=0; i<edge; i++){
        cin >> u >> v >> w; // bidirectional graph
        adj[u].push_back(v); adj[v].push_back(u);
        cost[u][v]=w; cost[v][u]=w;
    }

    mincst=primsMST(node, 1);

    cout <<endl<<"Minimum Spanning Tree:"<<endl<<endl;
    cout <<" Edges "<<"  Weight"<<endl;

    for(int i=1; i<=node-1; i++){
        u=Q.front(); Q.pop();
        v=Q.front(); Q.pop();
        w=cost[u][v];
        printf("%2d  %2d   %3d\n\n",u,v,w);
    }
    cout <<endl<<"Total Minimum Cost: "<<mincst<<endl;
    return 0;
}
```