Q1.

```
#include <bits/stdc++.h>
#define ll long long
#define INF INT_MAX
using namespace std;
int prims(int **arr, int n)
{
    vector<bool> visited(n, false);
    vector<int> weight(n, INF);
    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int,
int>>> min_heap;
    int src = 0;
    weight[src] = 0;
    min_heap.push(make_pair(weight[src], src));
    while (!min_heap.empty())
    {
        int u = min_heap.top().second;
        min_heap.pop();
        if (!visited[u])
        {
            visited[u] = true;
            for (int v = 0; v < n; ++v)
            {
                if (!visited[v] && arr[u][v] != 0 && arr[u][v] < weight[v])
                {
                    weight[v] = arr[u][v];
                    min_heap.push(make_pair(weight[v], v));
                }
            }
        }
    }

    int sum = 0;
    for (auto i : weight)
        sum += i;
    return sum;
}
int main()
{
    int n;
    cin >> n;
    int **arr;
    arr = (int **)malloc(n * sizeof(int *));
    for (int i = 0; i < n; ++i)
        arr[i] = (int *)malloc(n * sizeof(int));
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
```

```
            cin >> arr[i][j];
    cout << "Minimum spanning weight : " << prims(arr, n);
    return 0;
}
```

Output-

```
7

0 0 7 5 0 0 0

0 0 8 5 0 0 0

7 8 0 9 7 0 0

5 5 9 0 15 6 0

0 0 7 15 0 8 9

0 0 0 6 8 0 11

0 0 0 0 9 11 0


[Success] Your code was executed successfully
Minimum spanning weight : 39
```

Q2.

```
#include <algorithm>
#include <iostream>
#include <vector>
#include<queue>
using namespace std;
typedef pair<int,pair<int,int>>pi;
int find(int p[10],int x)
{
    if(p[x]==-1)
    return x;
    return find(p,p[x]);
}
void union1(int u,int v,int s[10],int p[10])// here sfor parent and p for
status
{
    int pu=find(s,u);
    int pv=find(s,v);
    if(pu!=pv)
    {
        if(p[pu]<p[pv])
        {
            s[pu]=pv;
            p[pv]=p[pv]+p[pu];

        }
        else
        {
            s[pv]=pu;
            p[pu]=p[pu]+p[pv];

        }

    }
}
void kruskal(priority_queue<pi,vector<pi>,greater<pi>>pq,int n,int p[10],int
st[10])
{

    int sum=0,i,j,w,pi,pj;
```

```
    while(!pq.empty())
    {
    w=pq.top().first;
    i=pq.top().second.first;
    j=pq.top().second.second;

    pq.pop();
    pi=find(p,i);
    pj=find(p,j);
    if(pi!=pj )
    {

    union1(i,j,p,st);
    sum+=w;
    }

    }
    cout<<sum;}
int main() {
    int c=0,i,j,n,key,p[10],st[10];
    priority_queue<pi,vector<pi>,greater<pi>>pq;
    cin>>n;
    for(i=0;i<n;i++)
    {

        for(j=0;j<n;j++)
        {
            cin>>key;
        if(key!=0)
        {
            pq.push(make_pair(key,make_pair(i,j)));
            c++;
        }
        st[i]=1;
        p[i]=-1;

        }

}

kruskal(pq,c,p,st);
    return 0;
}
```

Output-

```
7

0 0 7 5 0 0 0

0 0 8 5 0 0 0

7 8 0 9 7 0 0

5 5 9 0 15 6 0

0 0 7 15 0 8 9

0 0 0 6 8 0 11

0 0 0 0 9 11 0


[Success] Your code was executed successfully
39
```

Q3.

```
#include <algorithm>
#include <iostream>
#include <vector>
#include<queue>
using namespace std;
typedef pair<int,pair<int,int>>pi;
int find(int p[10],int x)
{
    if(p[x]==-1)
    return x;
    return find(p,p[x]);
}
void union1(int u,int v,int s[10],int p[10])// here sfor parent and p for
status
{
    int pu=find(s,u);
    int pv=find(s,v);
    if(pu!=pv)
    {
        if(p[pu]<p[pv])
        {
            s[pu]=pv;
            p[pv]=p[pv]+p[pu];
         }
        else
        {
            s[pv]=pu;
            p[pu]=p[pu]+p[pv];
            }}}
void kruskal(priority_queue<pi,vector<pi>>pq,int n,int p[10],int st[10])
{
    int sum=0,i,j,w,pi,pj;
    while(!pq.empty())
    {
    w=pq.top().first;
    i=pq.top().second.first;
    j=pq.top().second.second;
    pq.pop();
    pi=find(p,i);
```

```
    pj=find(p,j);
    if(pi!=pj )
    {
    union1(i,j,p,st);
    sum+=w;
    }

    }
    cout<<sum;}
int main() {
    int c=0,i,j,n,key,p[10],st[10];
    priority_queue<pi,vector<pi>>pq;
    cin>>n;
    for(i=0;i<n;i++)
    {

        for(j=0;j<n;j++)
        {
            cin>>key;
        if(key!=0)
        {
            pq.push(make_pair(key,make_pair(i,j)));
            c++;
        }
        st[i]=1;
        p[i]=-1;

        }

}

kruskal(pq,c,p,st);
    return 0;
}
```

Output-

```
7

0 0 7 5 0 0 0

0 0 8 5 0 0 0

7 8 0 9 7 0 0

5 5 9 0 15 6 0

0 0 7 15 0 8 9
```

```
00068011

00009110

[Success] Your code was executed successfully
59
```