

WEEK – 8

PRIYANSHI

2016927

E

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)

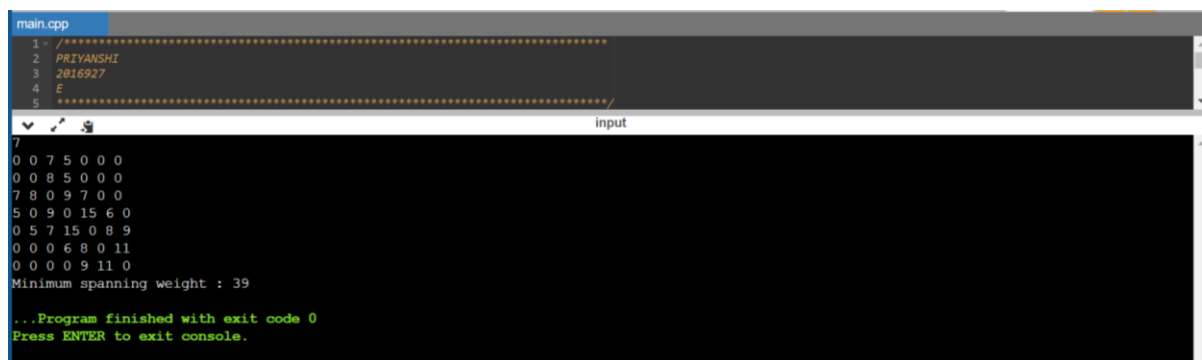
            cin >> arr[i][j];

    cout << "Minimum spanning weight : " << prims(arr, n);

    return 0;
}

```

OUTPUT



The screenshot shows a C++ program running in a terminal. The program reads a 7x7 adjacency matrix from standard input. The input data is as follows:

	0	1	2	3	4	5	6	7
0	0	7	5	0	0	0	0	0
1	0	0	8	5	0	0	0	0
2	7	8	0	9	7	0	0	0
3	5	0	9	0	15	6	0	0
4	0	5	7	15	0	8	9	0
5	0	0	0	6	8	0	11	0
6	0	0	0	0	9	11	0	0
7	0	0	0	0	0	0	0	0

The program outputs the minimum spanning weight as 39. The terminal window shows the following output:

```

7
0 0 7 5 0 0 0
0 0 8 5 0 0 0
7 8 0 9 7 0 0
5 0 9 0 15 6 0
0 5 7 15 0 8 9
0 0 0 6 8 0 11
0 0 0 9 11 0
Minimum spanning weight : 39
...Program finished with exit code 0
Press ENTER to exit console.

```

Q2.

```
#include <bits/stdc++.h>
```

```
#define NIL -1
```

```
using namespace std;
```

```
int findParent(vector<int> parent, int u)
```

```
{
```

```
    if (parent[u] < 0)
```

```
        return u;
```

```
    return findParent(parent, parent[u]);
```

```
}
```

```
bool UnionByWeight(vector<int> &parent, int u, int v)
```

```
{
```

```
    int pu = findParent(parent, u);
```

```
    int pv = findParent(parent, v);
```

```
    if (pu != pv)
```

```
    {
```

```
        if (parent[pu] <= parent[pv])
```

```
        {
```

```
            parent[pu] += parent[pv];
```

```
            parent[pv] = pu;
```

```
        }
```

```
    else
```

```
    {
```

```
        parent[pv] += parent[pu];
```

```

        parent[pu] = pv;
    }

    return true;
}

return false;
}

```

```

int kruskals(int **graph, int n)
{
    vector<pair<int, pair<int, int>>> G;
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            if (graph[i][j] != 0)
                G.push_back(make_pair(graph[i][j], make_pair(i, j)));
    sort(G.begin(), G.end());
    vector<int> parent(n, NIL);
    int s = 0;
    for (auto i : G)
    {
        int u = i.second.first;
        int v = i.second.second;
        int w = i.first;
        if (UnionByWeight(parent, u, v))

```

```

        s += w;

    }

    return s;
}

int main()
{
    int n;

    cin >> n;

    int **graph;

    graph = (int **)malloc(n * sizeof(int *));

    for (int i = 0; i < n; ++i)

        graph[i] = (int *)malloc(n * sizeof(int));

    for (int i = 0; i < n; ++i)

        for (int j = 0; j < n; ++j)

            cin >> graph[i][j];

    cout << "Minimum spanning weight : " << kruskals(graph, n) << endl;

    return 0;
}

```

OUTPUT

```
main.cpp
1- /*.....
2  PRIYANKHI
3  2016927
4  E
5  .....*/
6  #include <bits/stdc++.h>

input
7
0 0 7 5 0 0 0
0 0 8 5 0 0 0
7 8 0 9 7 0 0
5 0 9 0 15 6 0
0 5 7 15 0 8 9
0 0 0 6 8 0 11
0 0 0 0 9 11 0
Minimum spanning weight : 37

...Program finished with exit code 0
Press ENTER to exit console.
```

Q3.

```
#include <bits/stdc++.h>
```

```
#define NIL -1
```

```
using namespace std;
```

```
int findParent(vector<int> parent, int u)
```

```
{
```

```
    if (parent[u] < 0)
```

```
        return u;
```

```
    return findParent(parent, parent[u]);
```

```
}
```

```
bool UnionByWeight(vector<int> &parent, int u, int v)
```

```
{
```

```
    int pu = findParent(parent, u);
```

```
    int pv = findParent(parent, v);
```

```

if (pu != pv)
{
    if (parent[pu] <= parent[pv])
    {
        parent[pu] += parent[pv];
        parent[pv] = pu;
    }
    else
    {
        parent[pv] += parent[pu];
        parent[pu] = pv;
    }
    return true;
}
return false;
}

```

```

int kruskals(int **graph, int n)
{
    vector<pair<int, pair<int, int>>> G;
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            if (graph[i][j] != 0)

```



```

        G.push_back(make_pair(graph[i][j], make_pair(i, j)));
sort(G.begin(), G.end(), greater<pair<int, pair<int, int>>>());
vector<int> parent(n, NIL);
int s = 0;
for (auto i : G)
{
    int u = i.second.first;
    int v = i.second.second;
    int w = i.first;
    if (UnionByWeight(parent, u, v))
        s += w;
}
return s;
}

```

```

int main()
{
    int n;
    cin >> n;
    int **graph;
    graph = (int **)malloc(n * sizeof(int *));
    for (int i = 0; i < n; ++i)
        graph[i] = (int *)malloc(n * sizeof(int));
}

```

```
    for (int i = 0; i < n; ++i)

        for (int j = 0; j < n; ++j)

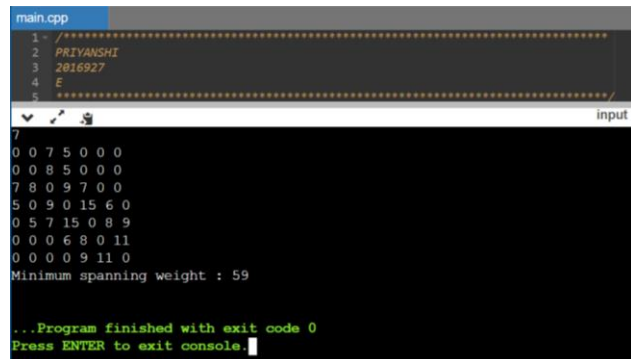
            cin >> graph[i][j];

    cout << "Minimum spanning weight : " << kruskals(graph, n) << endl;

    return 0;

}
```

OUTPUT



The screenshot shows a code editor with a file named 'main.cpp' containing a C++ program. The program reads an 8x8 adjacency matrix from standard input. The input data is as follows:

	0	1	2	3	4	5	6	7
0	0	7	5	0	0	0	0	0
1	0	0	8	5	0	0	0	0
2	7	8	0	9	7	0	0	0
3	5	0	9	0	15	6	0	0
4	0	5	7	15	0	8	9	0
5	0	0	0	6	8	0	11	0
6	0	0	0	0	9	11	0	0
7	0	0	0	0	0	0	0	0

The program outputs the minimum spanning tree weight as 59. The console window shows the following text:

```
Minimum spanning weight : 59

...Program finished with exit code 0
Press ENTER to exit console.
```