```cpp
#include <bits/stdc++.h>
using namespace std;
typedef pair<int, int> edgeWightPair;

class Graph

{
    int V;
    list<edgeWightPair> *adj;

public:
    Graph(int V)
    {
        this->V = V;
        adj = new list<edgeWightPair>[V];
    }
    void addEdge(int u, int v, int w)
    {
        adj[u].push_back(make_pair(v, w));
        adj[v].push_back(make_pair(u, w));
    }
    void printNeighbour(int chk)
    {
        cout << chk << " : ";
        for (auto i = adj[chk].begin(); i != adj[chk].end(); i++)
        {
            cout << "(" << (*i).first << " ," << (*i).second << ")";
        }
    }

    void BFS(int source)
    {

        vector<bool> visited(V, false);
        queue<int> Q;
        visited[source] = true;

        Q.push(source);
```

```
        while (!Q.empty())
        {
            int u = Q.front();
            cout << u << " ";
            Q.pop();

            for (auto element : adj[u])
            {
                int v = element.first;
                if (visited[v] != true)
                {
                    visited[v] = true;

                    Q.push(v);
                }
            }
        }
    }

    void DFS(int source)
    {

        static vector<bool> visited(V, false);
        queue<int> Q;
        visited[source] = true;

        cout << source << " ";

        for (auto element : adj[source])
        {
            int v = element.first;
            if (visited[v] != true)
            {
                DFS(v);
            }
        }
    }
};
int main()
{
```

```
    int V, E, source;
    cin >> V >> E >> source;
    Graph g(V);
    for (int i = 0; i < E; i++)
    {
        int v, w, u;
        cin >> u >> v >> w;
        g.addEdge(u, v, w);
    }

    for (int i = 0; i < V; i++)
    {
        g.printNeighbour(i);
        cout << endl;
    }

    cout << endl
         << endl;
    g.BFS(source);
    cout << endl
         << endl;
    g.DFS(source);

    return 0;
}

/*

7 10 0
0 1 7
0 2 1
0 5 3
1 3 11
2 3 3
3 6 1
6 5 2
6 4 4
5 4 4
2 5 8
```

```
*/
```