Name: Tirth Hihoriya

Roll no.: 18bce244

Prac-2 : Check Isomorphism

```
#include <bits/stdc++.h>
using namespace std;

class Graph {
public:
    set<char> V;
    set<pair<char,char>> E;

    Graph() {}
    Graph(set<char> Vertices, set<pair<char,char>> Edges)
    {
        V = Vertices;
        E = Edges;
    }
};


// # Print the vertices and edges of graph
void print_Graph(Graph G)
{
    cout << "\tVertices: { ";
        for (auto it = G.V.begin() ; it != G.V.end() ; ++it ) {
            cout << *it << " ";
        }
        cout << "}\n";
        cout << "\tEdges:    { ";
        for (auto it = G.E.begin() ; it != G.E.end() ; ++it ) {
            cout << "{" << it->first << ", " << it->second << "} ";
        }
        cout << "}\n\n";
}


// # Making adjacency list
map<char,set<char>> make_adj_list(set<pair<char,char>> E)
{
    map<char,set<char>> adj_list;
    for(auto it = E.begin() ; it != E.end() ; ++it ) {
        adj_list[it->first].insert(it->second);
        adj_list[it->second].insert(it->first);
    }
    return adj_list;
}
```

```
vector<pair<char,int>> degrees_of_vertices(set<pair<char,char>> E) {

    map<char,set<char>> adj_list = make_adj_list(E);
    vector<pair<char,int>> degree;
    for (auto p : adj_list) {
        auto vert = p.first;
        auto adj = p.second;
        degree.push_back(std::make_pair(vert, adj.size()));
    }
    return degree;
}

bool compare_func(pair<char,int> a, pair<char,int> b){
    return (a.second < b.second);
}


// # Checking isomorphism
bool is_isomorphic(Graph G1, Graph G2)
{
    set<char> V1 = G1.V;
    set<char> V2 = G2.V;

    set<pair<char,char>> E1 = G1.E;
    set<pair<char,char>> E2 = G2.E;

    if(!(V1.size() == V2.size()) && (E1.size() == E2.size())){
        return false;
    }
    else{
        vector<pair<char,int>> G1_degrees  = degrees_of_vertices(G1.E);
        sort(G1_degrees.begin(), G1_degrees.end(), compare_func);

        vector<pair<char,int>> G2_degrees  = degrees_of_vertices(G2.E);
        sort(G2_degrees.begin(), G2_degrees.end(), compare_func);

        // for(auto a: G1_degrees){
        //     cout << a.first << "-->" << a.second << '\n';
        // }

        for(int i=0;  i<G1_degrees.size(); i++){
            if ( G1_degrees[i].second != G2_degrees[i].second )
                return false;
        }

        cout << "Correspondence and Degrees\n";
        for(int i=0;  i<G1_degrees.size(); i++){
        cout << G1_degrees[i].first << " <---> " << G2_degrees[i].first <<
        "   D = " <<  G1_degrees[i].second  <<'\n';
        }

    }
    return true;
```

```
}


int main()
{
    set<char> V1 = {'a', 'b', 'c', 'd', 'e'};
    set<pair<char, char> > E1 = {{'a', 'b'}, {'a', 'c'}, {'b', 'c'}, {'c','d'},
{'b','d'},{'d','e'}};
    auto G1 = Graph(V1, E1);
    cout << "\nGraph  G1 : \n";
    print_Graph(G1);

    set<char> V2 = {'p', 'q', 'r', 's','t'};
    set<pair<char, char> > E2 = {{'p', 'q'}, {'p', 'r'}, {'p', 's'}, {'q','r'},
{'r','s'},{'s','t'}};
    auto G2 = Graph(V2, E2);
    cout << "Graph  G2 : \n";
    print_Graph(G2);


    bool result = is_isomorphic(G1,G2);
    if(result)
        cout << "\nResult :  G1 and G2 are isomorphic Graph\n\n";
    else
        cout << "\nResult :  G1 and G2 are not isomorphic Graph\n\n";
}
```

OUTPUT :

```
Graph  G1 :
        Vertices: { a b c d e }
        Edges:    { {a, b} {a, c} {b, c} {b, d} {c, d} {d, e} }

Graph  G2 :
        Vertices: { p q r s t }
        Edges:    { {p, q} {p, r} {p, s} {q, r} {r, s} {s, t} }

Correspondence and Degrees
e <---> t   D = 1
a <---> q   D = 2
b <---> p   D = 3
c <---> r   D = 3
d <---> s   D = 3

Result :  G1 and G2 are isomorphic Graph
```