Name: Tirth Hihoriya

Roll no.: 18bce244

Prac-1 : Operations on Graphs

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

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

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



// # Union
set<int> union_of_vertices(set<int> v1, set<int> v2)
{
    set<int> uni = v1;
    uni.insert(v2.begin(), v2.end());
    return uni;
}

set<pair<int,int>> union_of_edges(set<pair<int,int>> e1, set<pair<int,int>> e2)
{
    set<pair<int,int>> uni = e1;
    uni.insert(e2.begin(), e2.end());
    return uni;
}

Graph union_of_graph(Graph x, Graph y)
{
    set<int> V = union_of_vertices(x.V,y.V);
    set<pair<int,int>> E = union_of_edges(x.E,y.E);
    return Graph(V,E);
}



// # Intersection
```

```
set<int> intersection_of_vertices(set<int> v1, set<int> v2)
{
    set<int> uni = union_of_vertices(v1,v2);
    set<int> inter;

    for(auto it=uni.begin(); it!=uni.end(); it++)
    {
        if(v1.find(*it) != v1.end() && v2.find(*it) != v2.end())
        {
            inter.insert(*it);
        }
    }
    return inter;
}

set<pair<int,int>> intersection_of_edges(set<pair<int,int>> e1, set<pair<int,int>>
e2)
{
    set<pair<int,int>> uni = union_of_edges(e1,e2);
    set<pair<int,int>> inter;

    for(auto it=uni.begin(); it!=uni.end(); it++)
    {
        if(e1.find(*it) != e1.end() && e2.find(*it) != e2.end())
        {
            inter.insert(*it);
        }
    }
    return inter;
}

Graph intersection_of_graph(Graph x, Graph y){
    set<int> V = intersection_of_vertices(x.V,y.V);
    set<pair<int,int>> E = intersection_of_edges(x.E,y.E);
    return Graph(V,E);
}



// # Difference
set<pair<int,int>> difference_of_edges(set<pair<int,int>> e1, set<pair<int,int>>
e2)
{
    set<pair<int,int>> diff;
    for(auto it=e1.begin(); it!=e1.end(); it++)
    {
        if(e2.find(*it) == e2.end())
        {
            diff.insert(*it);
        }
    }
    return diff;
}
```

```
Graph difference_of_graph(Graph x, Graph y){
    set<int> V = x.V;
    set<pair<int,int>> E = difference_of_edges(x.E, y.E);
    return Graph(V,E);
}

Graph symmetric_difference_of_graph(Graph x, Graph y){
    set<int> V = union_of_vertices(x.V,y.V);

    set<pair<int,int>> e_uni = union_of_edges(x.E,y.E);
    set<pair<int,int>> e_inter =  intersection_of_edges(x.E,y.E);
    set<pair<int,int>> E = difference_of_edges(e_uni, e_inter);

    return Graph(V,E);
}


// # 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";
}


int main()
{
    set<int> V1 = {1, 2, 3, 4};
    set<pair<int, int> > E1 = {{1, 2}, {2, 3}, {3, 4}};
    auto G1 = Graph(V1, E1);

    set<int> V2 = {1, 2, 3};
    set<pair<int, int> > E2 = {{1, 3}, {2, 3}};
    auto G2 = Graph(V2, E2);


    int x;
    Graph G_union,G_intersect,G_difference;
    do{
        cout << "\n--> Menu for G1 and G2:\n";
        cout << "           1) Union of G1 and G2\n";
        cout << "           2) Intersection of G1 and G2\n";
        cout << "           3) Symmetric Difference of G1 and G2\n";
        cout << "           4) G1 - G2\n";
        cout << "           5) G2 - G1\n";
        cout << "           6) Print G1\n";
        cout << "            7) Print G2\n";
        cout << "            0) exit\n";


        cout<< "Enter your choice : ";
        cin>>x;

        switch(x){
            case 1:
                G_union = union_of_graph(G1,G2);
                cout << "Union :-\n";
                print_Graph(G_union);
                break;
            case 2:
                G_intersect = intersection_of_graph(G1,G2);
                cout << "Intersection :-\n";
                print_Graph(G_intersect);
                break;
            case 3:
                G_difference = symmetric_difference_of_graph(G1,G2);
                cout << "Symmetric difference :-\n";
                print_Graph(G_difference);
                break;
            case 4:
                G_difference = difference_of_graph(G1,G2);
                cout << "G1 - G2 :-\n";
                print_Graph(G_difference);
                break;
            case 5:
                G_difference = difference_of_graph(G2,G1);
                cout << "G2 - G1 :-\n";
                print_Graph(G_difference);
                break;
            case 6:
                cout << "G1 :-\n";
                print_Graph(G1);
                break;
            case 7:
                cout << "G2 :-\n";
                print_Graph(G2);
                break;
            case 0:
                break;
            default:
                cout << "Enter the vald number from the menu !!!\n";
        }
    }while(x!=0);
}
```

OUTPUT :

```


--> Menu for G1 and G2:
           1) Union of G1 and G2
           2) Intersection of G1 and G2
           3) Symmetric Difference of G1 and G2
           4) G1 - G2
           5) G2 - G1
           6) Print G1
           7) Print G2
           0) exit
Enter your choice : 6
G1 :-
        Vertices: { 1 2 3 4 }
        Edges:    { {1, 2} {2, 3} {3, 4} }


--> Menu for G1 and G2:
           1) Union of G1 and G2
           2) Intersection of G1 and G2
           3) Symmetric Difference of G1 and G2
           4) G1 - G2
           5) G2 - G1
           6) Print G1
           7) Print G2
           0) exit
Enter your choice : 7
G2 :-
        Vertices: { 1 2 3 }
        Edges:    { {1, 3} {2, 3} }


--> Menu for G1 and G2:
           1) Union of G1 and G2
           2) Intersection of G1 and G2
           3) Symmetric Difference of G1 and G2
           4) G1 - G2
           5) G2 - G1
           6) Print G1
           7) Print G2
           0) exit
Enter your choice : 1
Union :-
        Vertices: { 1 2 3 4 }
        Edges:    { {1, 2} {1, 3} {2, 3} {3, 4} }


--> Menu for G1 and G2:
           1) Union of G1 and G2
           2) Intersection of G1 and G2
           3) Symmetric Difference of G1 and G2
           4) G1 - G2
           5) G2 - G1
           6) Print G1
           7) Print G2
           0) exit
Enter your choice : 2
Intersection :-
        Vertices: { 1 2 3 }
        Edges:    { {2, 3} }


--> Menu for G1 and G2:
           1) Union of G1 and G2
           2) Intersection of G1 and G2
           3) Symmetric Difference of G1 and G2
           4) G1 - G2
           5) G2 - G1
           6) Print G1
           7) Print G2
           0) exit
Enter your choice : 3
Symmetric difference :-
        Vertices: { 1 2 3 4 }
        Edges:    { {1, 2} {1, 3} {3, 4} }


--> Menu for G1 and G2:
           1) Union of G1 and G2
           2) Intersection of G1 and G2
           3) Symmetric Difference of G1 and G2
           4) G1 - G2
           5) G2 - G1
           6) Print G1
           7) Print G2
           0) exit
Enter your choice : 4
G1 - G2 :-
        Vertices: { 1 2 3 4 }
        Edges:    { {1, 2} {3, 4} }


--> Menu for G1 and G2:
           1) Union of G1 and G2
           2) Intersection of G1 and G2
           3) Symmetric Difference of G1 and G2
           4) G1 - G2
           5) G2 - G1
           6) Print G1
           7) Print G2
           0) exit
Enter your choice : 5
G2 - G1 :-
        Vertices: { 1 2 3 }
        Edges:    { {1, 3} }


--> Menu for G1 and G2:
           1) Union of G1 and G2
           2) Intersection of G1 and G2
           3) Symmetric Difference of G1 and G2
           4) G1 - G2
           5) G2 - G1
           6) Print G1
           7) Print G2
           0) exit
Enter your choice : 6
G1 :-
        Vertices: { 1 2 3 4 }
        Edges:    { {1, 2} {2, 3} {3, 4} }


--> Menu for G1 and G2:
           1) Union of G1 and G2
           2) Intersection of G1 and G2
           3) Symmetric Difference of G1 and G2
           4) G1 - G2
           5) G2 - G1
           6) Print G1
           7) Print G2
           0) exit
Enter your choice : 0
```