**Agnim Gupta**

**2028083**

**A-23,CSSE**

**Question 1**

#include <iostream>

#include <vector>

#include <unordered\_map>

using namespace std;

class DisjointSet

{

 unordered\_map<int, int> parent;

public:

void makeSet(vector <int> const &universe)

{

   for (int i: universe)

   {

     parent[i] = i;

   }

}

int Find(int k)

{

   if (parent [k] == k) {

   return k;

   }

   return Find (parent [k]);

}

void Union(int a, int b)

 {

  int x = Find (a);

  int y = Find (b);

  parent [x] = y;

 }

};

void printSets (vector<int> const &universe, DisjointSet &ds)

{

for(int i: universe) {

  cout << ds. Find(i) << " ";

 }

 cout<< endl;

}

int main()

{

 vector<int> universe = { 1, 2, 3, 4, 5 };

 DisjointSet ds;

 ds.makeSet(universe);

 printSets (universe, ds);

 ds.Union (4, 3);

 printSets (universe, ds);

 ds.Union (2, 3);

 printSets (universe, ds);

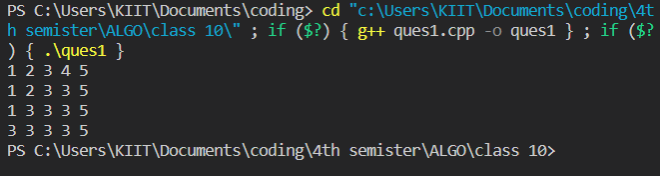
 ds.Union (1, 3);

 printSets (universe, ds);

 return 0;

}

**Output**

****

**Question 2**

#include <stdio.h>

#include <stdlib.h>

struct Edge

{

    int src, dest;

};

struct Graph

{

    int V, E;

    struct Edge\* edge;

};

struct subset

{

    int parent;

    int rank;

};

struct Graph\* createGraph(int V, int E)

{

    struct Graph\* graph = (struct Graph\*)malloc(sizeof(struct Graph));

    graph->V = V;

    graph->E = E;

    graph->edge = (struct Edge\*)malloc(graph->E \* sizeof(struct Edge));

    return graph;

}

int find(struct subset subsets[], int i)

{

    if (subsets[i].parent != i)

    subsets[i].parent = find(subsets, subsets[i].parent);

    return subsets[i].parent;

}

void Union(struct subset subsets[], int xroot, int yroot)

{

    if (subsets[xroot].rank < subsets[yroot].rank)

    subsets[xroot].parent = yroot;

    else if(subsets[xroot].rank > subsets[yroot].rank)

    subsets[yroot].parent = xroot;

    else

    {

        subsets[yroot].parent = xroot;

        subsets[xroot].rank++;

    }

}

int isCycle(struct Graph\* graph)

{

    int V = graph->V;

    int E = graph->E;

    struct subset\* subsets = (struct subset\*)malloc(V \* sizeof(struct subset));

    for (int v = 0; v < V; ++v)

    {

        subsets[v].parent = v;

        subsets[v].rank = 0;

    }

    for (int e = 0; e < E; ++e)

    {

        int x = find(subsets, graph->edge[e].src);

        int y = find(subsets, graph->edge[e].dest);

        if (x == y)

        return 1;

        Union(subsets, x, y);

    }

return 0;

}

int main()

{

int V = 3, E = 3;

struct Graph\* graph = createGraph(V, E);

graph->edge[0].src = 0;

graph->edge[0].dest = 1;

graph->edge[1].src = 1;

graph->edge[1].dest = 2;

graph->edge[2].src = 0;

graph->edge[2].dest = 2;

if (isCycle(graph))

{

    printf("Graph contains cycle");

}

else

{

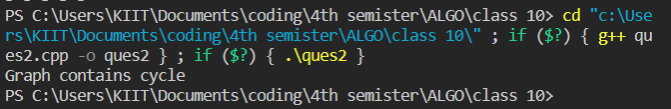
    printf("Graph doesn't contain cycle");

}

return 0;

}

**Output**

****