Assignment 4

Instructions:

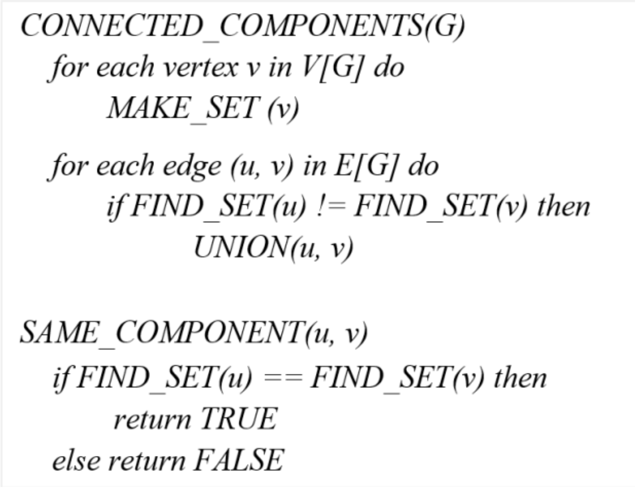
- Read the questions very carefully and write all the functions as you are instructed in the question. You should take the input following the sample input format. Your output should also match the sample outputs. Note that your code should work for all reasonable inputs, not just sample inputs.
- Adopting any unfair means will result in -100%.
- Submit the codes in ELMS. Name the files 1.cpp, 2.cpp etc. Only submit the .cpp files.

Question 1:

Complete the code of the attached file ***disjoint-set_incomplete.cpp***. **[10% marks]**

Question 2:

Implement the following algorithms for identifying the connected components of a graph and complete the codes of the attached file ***connected_comp_incomplete.cpp***. **[50% marks]**

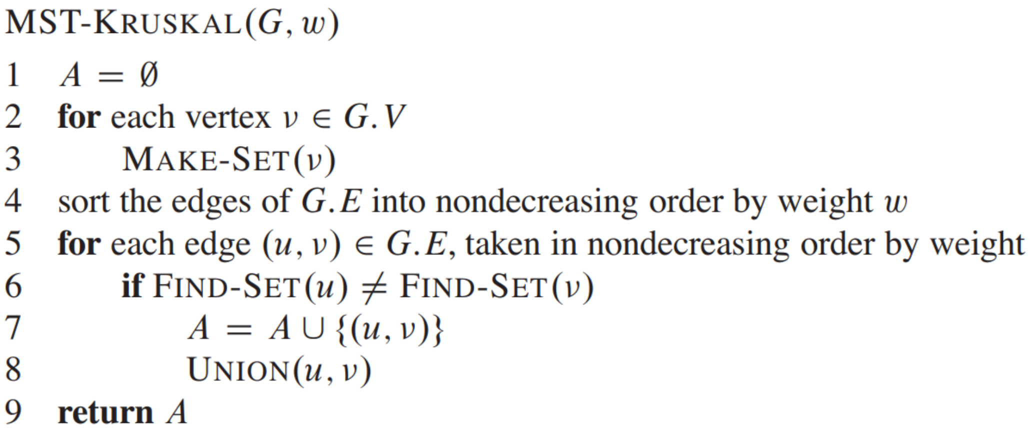


```
CONNECTED_COMPONENTS(G)
   for each vertex v in V[G] do
        MAKE_SET (v)

   for each edge (u, v) in E[G] do
        if FIND_SET(u) != FIND_SET(v) then
              UNION(u, v)

SAMPLE_COMPONENT(u, v)
   if FIND_SET(u) == FIND_SET(v) then
        return TRUE
   else return FALSE
```

Sample Input **V** **E** **u v** **…**	**Sample Output** **// this is a comment, not part of input or output**
8 **5** **0 1** **1 2** **0 3** **1 3** **4 5** **5 // option** **1 // option** **8** **1 // option** **7** **2 // option** **0 4** **3 // option** **0 4** **4 // option** **6 // terminate**	option 5: vertices of each connected components **// the order may be different for you** 0 1 2 3 4 5 6 7 option 1: print the root 8 is not an element of the disjoint set option 1: print the root root of 7 is 7 option 2: same component or not 0 and 4 are not in the same component option 3: there is a path or not there is no path between 0 and 4 option 4: print all the roots 0 4 6 7 **// these values may differ depending on your implementation**

Question 3

Implement the **following** algorithm for finding the Minimum Spanning Tree in **an *undirected weighted* graph**. **You must implement the disjoint set yourself using path-compression and union-by-rank heuristic. You must take input from the user.** **[60% marks]**



```
MST-KRUSKAL($G, w$)
$A = \emptyset$
for each vertex $v \in G.V$
    MAKE-SET($v$)
sort the edges of $G.E$ into nondecreasing order by weight $w$
for each edge $(u, v) \in G.E$, taken in nondecreasing order by weight
    if FIND-SET($u$) $\neq$ FIND-SET($v$)
        $A = A \cup \{(u, v)\}$
        UNION($u, v$)
return $A$
```

Sample Input **V** **E** **u v weight(u,v)** **…**	**Sample Output**

6
9
0 1 4
1 2 5
0 3 10
1 3 3
1 5 6
1 4 7
2 4 1
3 5 9
4 5 8
	MST
	2 – 4
	1 – 3
	0 – 1
	1 – 2
	1 – 5
	Weight: 1+3+4+5+6 = 19

4
5
0 1 10
0 2 6
0 3 5
1 3 15
2 3 4
	MST
	2 – 3
	0 – 3
	0 – 1
	Weight: 4+5+10 = 19