1. Answer each of the following questions as True or False.
   (a) A MST contains a cycle. **False**
   (b) If we remove an edge from a MST, the resulting subgraph is still a MST. **False**
   (c) If we add an edge to a MST, the resulting subgraph is still a MST. **False**
   (d) If there are V vertices in a given graph, a MST of that graph contains $|V| - 1$ edges. **True**
   (e) Every MST is a sparse graph. **True**

2. Following is the pseudocode for Kruskal’s algorithm to find a MST.

   ```python
   def Kruskal(Graph G):
       initialize each vertex to be a component
       sort all edges by weight
       for each edge (u, v) in sorted order do
           if u and v are in different components then
               add edge (u,v) to the MST
               update u and v to be in the same component
           end if
       end for
   end function
   ```

   (a) Execute Kruskal’s algorithm on the following graph. Fill the table.

   (b) In this graph there are 6 vertices and 11 edges, and the for loop in the above pseudocode iterates 11 times, a few more times after the MST is found. How would you optimize the pseudocode so the for loop terminates early, as soon as a valid MST is found. Annotate the given pseudocode to add/edit lines.

   **Terminate for loop early, when MST has $|V| - 1$ edges.**
Figure 1: Disjoin-set. Rank of trees in the forest (from left): 2, 0, 2, 1.

3. Consider the disjoin-set shown in Figure 1. What would be the result of the following calls on union if we add the “union-by-rank” optimization. Draw the forest at each stage with corresponding ranks for each tree.
   i. union(2, 13)
   ![Diagram i. union(2, 13)]
   
   ii. union(4, 12)
   ![Diagram ii. union(4, 12)]
   
   iii. union(2, 8)
   ![Diagram iii. union(2, 8)]