Problem 1

Pseudo-code

```
IsMST(G: Graph, T: Tree):
    treeWeight = All tree edge weights summed
    treeVertices = All tree vertices
    graphVertices = All graph vertices
    if treeVertices != graphVertices   # The MST includes
                                         all vertices
        return

    MST = Kruskal(G)
    weight = MST.GetWeight()
    if treeWeight = weight   # MST from kruskal returns
                               minimum weight of graph
        return True            The T weight has to equa
                               the same weight to be a
    else                       MST
        return False
```

In order for the tree T to be an MST, T has to have the same weight as the MST generated from the Kruskal algorithm. So if the weight of T is equal to the weight of the MST weight, T is therefore an MST, as Kruskal algorithm always generates a MST. Algorithm time complexity would be $O(|E| \log |E|)$

Problem 2

Pseudo-code

```
Kruskal (G: Graph):
    MST = new Graph()

    Sort G edges' weight ascendingly

    while edge < G's # of vertices - 1 :
        get V1, V2, weight from edge of G
        If parent of V1 != parent of V2 :
            MST.AddEdge (V1, V2, weight)
            edge += 1
            Union (V1, V2)

    return MST
```

Kruskal algoritm would take most $O(|E| \log|E|)$. The sorting of G's edges would be $O(|E| \log|E|)$. Finding the parent of V1 and V2 and union operations would take $O(\log|E|)$. The algorithms purpose is to find a MST. It does so by piecing the smallest weighted edges together to connect to all vertices. While doing so it creates child to parent relationships through the union function.