Please submit one assignment per group.

1. (a) Suppose that the edge costs are all distinct in an input G to the MST problem. Prove that G has a unique minimum cost spanning tree. Do this by considering the proof that Kruskal's algorithm always finds a minimum cost spanning tree.

   (b) Now suppose that all edge costs are distinct, except $c_1 = c_2$, where $c_1 = c(e_1)$ and $c_2 = c(e_2)$ are the two minimum edge costs. Prove or give a counter-example: G has a unique minimum cost spanning tree.

   (c) Finally suppose that all edge costs are distinct, except $c_1 = c_2 = c_3$, where $c_1 = c(e_1)$, $c_2 = c(e_2)$, $c_3 = c(e_3)$ are the three minimum edge costs. Prove or give a counter-example: G has a unique minimum cost spanning tree.

2. The Make-Change Problem consists in paying a given amount using the least number of coins, using some fixed denominations, and an unlimited supply of coins of each denomination. Consider the following greedy algorithm to solve the change making problem, where the denominations are $\{1, 10, 25, 100\}$:

```
Make Change($N$) (makes change for $N$ units using coins from $C$)
    const $C \leftarrow \{1, 10, 25, 100\}$
    $L \leftarrow \emptyset$ ($L$ is the list holding the solution)
    $s \leftarrow 0$ ($s$ is the sum of the items in $L$)
    while $s < N$ do
        $x$ = largest item in $C$ such that $s + x \leq N$
        $L \leftarrow L, x$
        $s \leftarrow s + x$
    return $L$
```

   (a) Show that this greedy algorithm (with the given denominations) does *not* necessarily produce an optimal solution. That is, present an N for which the output L of MakeChange(N) contains more coins than the optimal solution.

   (b) Suppose now that the available denominations are $\{1, p, p^2, \ldots, p^n\}$, where $p > 1$ and $n \geq 0$ are integers. That is, "$C \leftarrow \{1, 10, 25, 100\}$" in the above algorithm is replaced by "$C \leftarrow \{1, p, p^2, \ldots, p^n\}$." Show that with this series of denominations (for some fixed p, n) the greedy algorithm above always finds an optimal solution.

   (c) Implement the Make Change algorithm in Python.