1. Prove the following statement.

When using the union-by-rank data structure, any root node of rank $k$ has at least $2^k$ nodes in the tree.

Solution: We prove this using induction on the rank $k$.

Base case: $k = 0$ the number of node in the tree is 1, which is no smaller than $2^0$.

Inductive assumption: Assume true for any root node of rank 0, 1, ..., $k - 1$

Inductive step:

For any root node of rank $k$, consider the process that it is formed. A rank $k$ root must be created by pointing a rank $k - 1$ root to another $k - 1$ root. These two roots have rank $k - 1$, so the inductive assumption applies and they each contain at least $2^{k-1}$ nodes. After merging, the total nodes contained in the rank $k$ root tree will be at least $2^{k-1} + 2^{k-1} = 2^k$.

This completes the proof.

2. Consider the Change Problem in Austria. The input to this problem is an integer $L$. The output should be the minimum cardinality collection of coins required to make $L$ shillings of change (that is, you want to use as few coins as possible). In Austria the coins are worth 1, 5, 10, 20, 25, 50 Shillings. Assume that you have an unlimited number of coins of each type. Formally prove or disprove that the greedy algorithm (that takes as many coins as possible from the highest denominations) correctly solves the Change Problem. So for example, to make change for 234 Shillings the greedy algorithms would take four 50 shilling coins, one 25 shilling coin, one 5 shilling coin, and four 1 shilling coins.

This algorithm is not optimal. For example, to make change of 40 Shillings, the algorithm would use 3 coins (25, 10, 5), where it is possible to use 2 20-Shilling coins.

3. Textbook 5.8
solution: Consider the edge \( e = (s, u) \) that has the minimum weight among all the edges incident on \( s \). Since all the edge weights are positive, this is the unique shortest path from \( s \) to \( u \) and hence must be present in any shortest path tree. Also, this edge is the unique cheapest edge across the cut \( \{s\} \) vs. \( V \setminus \{s\} \). Thus we know every MST must contain this edge. So all the shortest path trees and any MST must share at least one edge.

4. Textbook 5.14
a \( a \leftarrow 0, b \leftarrow 10, c \leftarrow 110, d \leftarrow 1110, e \leftarrow 1111. \)
b \( \text{length} = \frac{1000000}{2} \times 1 + \frac{1000000}{4} \times 2 + \frac{1000000}{8} \times 3 + \frac{1000000}{16} \times 4 + \frac{1000000}{4} \times 4 = 1875000 \)

5. Textbook 5.15
a \( f_a = 1/2, f_b = 1/4, f_c = 1/4 \) gives this code.
b This is not a prefix free code.
c This is not a full binary tree, thus cannot be an optimal code.

6. Textbook 5.16
a Let \( x \) be the symbol with the highest frequency with \( f(x) > 2/5 \) and suppose that it merges with some other symbol \( y \) during the process of constructing the tree and hence does not correspond to a codeword of length 1. To be merged, the nodes \( x \) and \( y \) must be the two least frequent symbol. This means there was at least one other node \( z \) (formed by merging other nodes, otherwise it will have a code of length 1), with \( f(z) > f(x) \) and \( f(z) > f(y) \). Thus, \( f(z) > 2/5 \) and hence \( f(y) < 1/5 \). Now let’s consider the two nodes that merged into \( z \), let’s call them \( v \) and \( w \). We know that \( f(v) + f(w) = f(z) > 2/5 \), thus at least one of \( v \) and \( w \) would have frequency greater than 1/5. Now this is not possible, because \( f(y) < 1/5 \) so \( v \) and \( w \) could not have been the two least frequent symbols.

b Let’s assume for contradiction that this is not true. Let \( x \) be a node corresponding to a single character with \( p(x) < 1/3 \) and the code for \( x \) is 1 bit. Then \( x \) must not be merged with any other nodes till the last step (in the binary tree). Consider the stage when there are only three nodes \( x, y, z \) left. Because the frequency of \( x \) is less than 1/3, the combined frequency of \( y \) and \( z \) must be greater than 2/3. Thus at one of them (let’s say it is \( y \)) will have frequency that is greater than 1/3, thus greater than frequency of \( x \). Now this tree cannot possibly be optimal because switching the positions of \( y \) with \( x \) in this tree will lead to a lower cost. This leads to a contradiction.