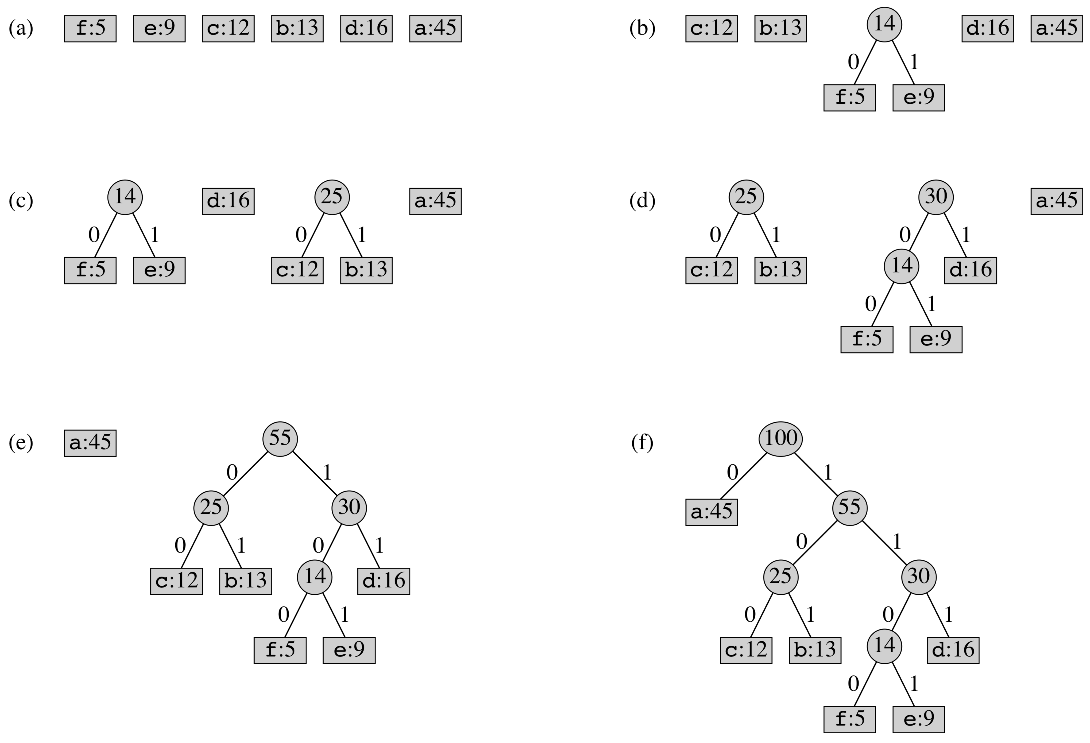
# Exercises, chapter 3, solutions

1. The answer is 128.

   **Explanation:** Each bit has two possible values. With 7 bits, $\overbrace{2\cdot 2\cdots 2}^{7} = 2^7 = 128$ different numbers can be represented, and each such number can encode one unique character.

2. See Chapter 3.1 in the textbook or the slides from Lecture 3.

3. (a) We apply the Huffman coding technique to obtain a variable-length encoding as follows.



Thus, the variable-length encoding is:

| Character | a | b | c | d | e | f |
|---|---|---|---|---|---|---|
| Encoding | 0 | 101 | 100 | 111 | 1101 | 1100 |

**Remark:** The above figure is from:

- T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein: *Introduction to Algorithms (Third Edition)*, The MIT Press, 2009. [Figure 16.5]

(b) The total number of bits becomes $45,000 \cdot 1 + 13,000 \cdot 3 + 12,000 \cdot 3 + 16,000 \cdot 3 + 9,000 \cdot 4 + 5,000 \cdot 4 = 224,000$, which gives a savings of about 25% compared to the fixed-length encoding.

4. One solution is to extract a smallest remaining element from the input lists and place it at the end of the output list, and repeat this step until all the input lists have become empty.

   The time complexity of this approach will depend on how quickly we can find a smallest remaining element. According to the problem specifications, each of the input lists is sorted, which means that we just need to choose among the current heads of the lists every time. However, if we simply examine all the current heads of the $k$ input lists in each of the $n$ iterations, the time complexity will be $O(nk)$. To do it more efficiently, we can store the current heads in a priority queue. The details are given below.

   **Algorithm:**

```
• Create an empty list $L$ and an empty priority queue $Q$ using the CreatePQ operation.
  For every input list $L_i$, let $x$ be the first element in $L_i$ and do InsertInPQ($Q, x$) (note that
  each such $x$ is contained in both $L_i$ and $Q$ at this point).
• Repeat the following until SizePQ($Q$) $= 0$:
    – Do ExtractMinFromPQ($Q$) to find and remove a smallest element $x$ from $Q$.
    – Let $y$ be the successor of $x$ in the input list $L_i$ that $x$ originally came from.
      If $y \neq$ NULL, i.e., if $x$ is not the last element in $L_i$, then do InsertInPQ($Q, y$).
    – Remove $x$ from $L_i$ and insert a new element at the end of $L$ whose data is GetData($x$).
• Return $L$.
```

   **Time complexity analysis:**
   The algorithm performs $n$ `InsertInPQ` operations and $n$ `ExtractMinFromPQ` operations. The priority queue $Q$ always contains at most $k$ elements, so every `InsertInPQ` and `ExtractMinFromPQ` operation takes $O(\lg k)$ time. All other operations take $O(1)$ time for each of the $n$ elements. Thus, the total time complexity is $O(n \cdot \lg k \, + \, n \cdot \lg k \, + \, n \cdot 1) \, = \, O(n \lg k)$.