Question 1

• Design an $O(n)$ algorithm that, given a list of $n$ elements in array $A$, finds all the elements that appear more than $\frac{n}{2}$ times in the list.
• Then, design an algorithm that, given a list of $n$ elements, finds all the elements that appear more than $\frac{n}{4}$ times.
Question 2

You are given an infinite array $A[\cdot]$ in which the first $n$ cells contain integers in sorted order and the rest of the cells are filled with $\infty$. You are not given the value of $n$. Describe an algorithm that takes an integer $x$ as input and finds a position in the array containing $x$, if such a position exists, in $O(\log n)$ time. (If you are disturbed by the fact that the array $A$ has infinite length, assume instead that it is of length $n$, but that you don’t know this length, and that the implementation of the array data type in your programming language returns the error message $\infty$ whenever elements $A[i]$ with $i > n$ are accessed.)
In the United States, coins are minted with denominations of 1, 5, 10, 25 and 50 cents. Provide an algorithm that will enable to make change of \( n \) units using the minimum number of coins. Prove its correctness and analyze its time complexity.

Show that greedy algorithm doesn’t always give the minimum number of coins in a country whose denominations are \{1,6,10\}
Question 3

• **Solution (Greedy Approach):**

  **input:** number of units to make change for \( n \)

  **Output:** number of half dollars, quarters, dimes, nickels and pennies to use \((c_{50}, c_{25}, c_{10}, c_5, c_1)\)

  **Algorithm:** MakeChange\((n)\)
  
  \[
  c_{50} = \frac{n}{50} \\
  n = n \mod 50 \\
  c_{25} = \frac{n}{25} \\
  n = n \mod 25 \\
  c_{10} = \frac{n}{10} \\
  n = n \mod 10 \\
  c_5 = \frac{n}{5} \\
  n = n \mod 5 \\
  c_1 = n \\
  \]

  Return \((c_{50}, c_{25}, c_{10}, c_5, c_1)\)
Question 3

Assume that:

\[
\begin{align*}
\ell & = 50c_{50} + 25c_{25} + 10c_{10} + 5c_5 + c_1 \quad (\text{Greedy approach}) \\
\ell & = 50b_{50} + 25b_{25} + 10b_{10} + 5b_5 + b_1 \quad (\text{Best solution})
\end{align*}
\]

We want to show that:

\[
c_{50} + c_{25} + c_{10} + c_5 + c_1 \leq b_{50} + b_{25} + b_{10} + b_5 + b_1
\]

Since the best solution is not greedy at some points there will be fewer coins of some domination in the best solution vs. the greedy solution. We will show that any combination of coins with lower denominations which make up for the difference could be replaced with fewer coins.
Question 3

\[
\begin{align*}
\begin{cases}
  n &= 50c_{50} + 25c_{25} + 10c_{10} + 5c_5 + c_1 \quad (\text{Greedy approach}) \\
  n &= 50b_{50} + 25b_{25} + 10b_{10} + 5b_5 + b_1 \quad (\text{Best solution})
\end{cases}
\end{align*}
\]

If \(b_{50} < c_{50}\) then, \(25b_{25} + 10b_{10} + 5b_5 + b_1 \geq 50\), possibilities are:
1. If \(b_{25} \geq 2\), can be replaced with one half dollar.
2. If \(b_{25} = 1\), we must have one of the following combinations: (2 dimes, 1 nickel) OR (1 dime, 3 nickels) OR etc. any of these combinations can be replaced with one half dollar.
3. If \(b_{25} = 0\), we must also have one of the following combinations: (5 dimes) OR (4 dimes, 2 nickels) or any of these combination which can be replaced by half dollar.
Question 3

\[
\begin{align*}
  n &= 50c_{50} + 25c_{25} + 10c_{10} + 5c_5 + c_1 \quad \text{(Greedy approach)} \\
  n &= 50b_{50} + 25b_{25} + 10b_{10} + 5b_5 + b_1 \quad \text{(Best solution)}
\end{align*}
\]

If \( b_{50} = c_{50} \) and \( b_{25} < c_{25} \), \( 10b_{10} + 5b_5 + b_1 \geq 25 \), possibilities are:

1. If \( b_{10} \geq 3 \), can be replaced with (1 quarter, 1 nickel)

2. If \( b_{10} = 2 \), we must also have one of the following combinations: (1 nickel) OR (5 pennies) OR etc. which can be replaced by 1 quarter.

3. If \( b_{10} = 1 \), we must have one of the following combinations: (3 nickels) OR (2 nickels, 5 pennies) OR etc. which can be replaced by 1 quarter.

4. If \( b_{10} = 0 \), we must have one of the following combinations: (5 nickels) OR etc. all of which can be replaced by one quarter.

Entire proof would continue through the case if \( b_{25} = c_{25}, b_{10} = c_{10}, b_5 = c_5 \) ...
The *square* of a matrix $A$ is its product with itself, $AA$.

(a) Show that five multiplications are sufficient to compute the square of a $2 \times 2$ matrix.

(b) What is wrong with the following algorithm for computing the square of an $n \times n$ matrix?

“Use a divide-and-conquer approach as in Strassen’s algorithm, except that instead of getting 7 subproblems of size $n/2$, we now get 5 subproblems of size $n/2$ thanks to part (a). Using the same analysis as in Strassen’s algorithm, we can conclude that the algorithm runs in time $O(n^{\log_2 5})$.”