Problem 1 (20 points)

a. Consider the following simple procedure:
   It takes as input an Array A of n elements, and an integer k.
   The first call is with \( k = n \).

\[
\text{BrainDead}(A, k) \\
\quad \text{if } (k \leq 1) \\
\quad \quad \text{return} \\
\quad \quad \text{print elements of } A \text{ from 1 to } k \\
\quad \text{BrainDead}(A, (2/3)\times k)
\]

What is the recurrence relation for the time complexity of the algorithm?

b. (10 points) [Extra Credit - see above note]
   The algorithm is modified so that half of the time it returns just after printing, instead of calling itself recursively. What is the recurrence relation for the revised algorithm?
   \textbf{Hint:} There are two cases, with the total time being the weighted sum of the cost of the two cases
   \textbf{Further Hint:} The weights sum up to 1

Problem 2 (20 points)

Let \( f(n) \) and \( g(n) \) be two monotonically increasing functions. Prove the following statements:
(you cannot simply quote the textbook here)

(i) \( f(n) = O(g(n)) \) does not imply \( g(n) = O(f(n)) \)
   \textbf{Hint:} give a counter-example

(ii) \( f(n) = O(g(n)) \) implies \( g(n) = \Omega(f(n)) \)
   \textbf{Hint:} prove using \( c, n_0 \) given in the definition of big-Oh
Problem 3 (20 points)

Write the solution to the following recurrences using the big-Oh notation. Make a statement (less than 3 lines) explaining the reasoning behind your guess. You do not have to derive/prove your solution.

\[
   (i) \quad T(n) = \begin{cases} 
   2T(\frac{n}{2}) + 1 & \text{if } n > 1 \\
   1 & \text{if } n = 1 
\end{cases}
\]

\[
   (ii) \quad T(n) = \begin{cases} 
   2T(\frac{n}{2}) + n & \text{if } n > 1 \\
   1 & \text{if } n = 1 
\end{cases}
\]

\[
   (iii) \quad T(n) = \begin{cases} 
   T(\sqrt{n}) + 1 & \text{if } n > 2 \\
   1 & \text{if } n \leq 2 
\end{cases}
\]

Problem 4 (20 points)

Solve any 2 of the following recurrences using any method of your choice:

\[
   (i) \quad T(n) = \begin{cases} 
   4T(\frac{n}{4}) + n & \text{if } n > 3 \\
   1 & \text{if } n \leq 3 
\end{cases}
\]

\[
   (ii) \quad T(n) = \begin{cases} 
   T(\frac{n}{2}) + T(\frac{n}{3}) + n & \text{if } n > 30 \\
   10n & \text{if } n \leq 30 
\end{cases}
\]

\[
   (iii) \quad T(n) = \begin{cases} 
   T(n - 1) + \log n & \text{if } n > 2 \\
   1 & \text{if } n \leq 2 
\end{cases}
\]

\[
   (iv) \quad T(n) = \begin{cases} 
   \sqrt{n}T(\sqrt{n}) + n \log n & \text{if } n > 2 \\
   1 & \text{if } n \leq 2 
\end{cases}
\]

Problem 5 (20 points)

Let S be an array of n (not necessarily distinct) integers. Describe an \(O(n)\) time algorithm to test whether any item occurs more than \([n/2]\) times in S.

*Hint: We know selection can be done in linear time*

Extra Credit (20 points)

Let S be a set of n integers. Describe an \(O(n^2 \log n)\) algorithm to determine whether there exists three integers \(a, b, c \in S\) (\(a, b, c\) need not be distinct) such that \(a + b + c = 0\)