Problem 1 (5 marks). Let \( f(n) \) be a function of positive integer \( n \). We know:
\[
\begin{align*}
  f(1) &= 1 \\
  f(n) &= 10 + 2 \cdot f(\lceil n/8 \rceil).
\end{align*}
\]
Use the Master Theorem to prove \( f(n) = O(n^{1/3}) \).

Solution. \( \alpha = 2, \beta = 8, \gamma = 0 \). We know: \( \log_\beta \alpha = 1/3 > \gamma \). Hence, \( f(n) = O(n^{\log_\beta \alpha}) = O(n^{1/3}) \).

Problem 2 (7 marks). Prove \( 5n + 3\sqrt{n} = O(n) \).

Solution. \( 5n + 3\sqrt{n} \leq 8n \) for all \( n \geq 1 \).

Problem 3 (8 marks). You are given a positive integer \( n \geq 3 \). Design an algorithm to determine whether \( n \) is a prime number. Your algorithm should have a cost of \( O(\sqrt{n}) \). Remember that it is not an atomic operation to calculate \( \sqrt{n} \).

Solution. Set \( x = 2 \). Repeat the following until \( x^2 > n \): (i) test whether \( n/x \cdot x = n \); (ii) if so, return “no”; otherwise, increase \( x \) by 1. When \( x^2 \) exceeds \( n \), return “yes”.

Problem 4 (15 marks). Let \( S_1 \) and \( S_2 \) be two sets of integers (they are not necessarily disjoint). We know that \( |S_1| = |S_2| = n \) (i.e., each set has \( n \) integers). Design an algorithm to report the distinct integers in \( S_1 \cup S_2 \) using \( O(n \log n) \) time. For example, if \( S_1 = \{1, 5, 6, 9, 10\} \) and \( S_2 = \{5, 7, 10, 13, 15\} \), you should output: 1, 5, 6, 7, 9, 10, 13, 15.

Solution. Put together \( S_1 \) and \( S_2 \), and sort the multi-set. Let \( L \) be the sorted list. For each \( i \in [1, 2n] \), output \( L[i] \) if (i) \( i = 1 \), or (ii) \( L[i] \neq L[i-1] \).

Problem 5 (10 marks). Suppose that we use binary search to find 90 in the sorted array \( A = (5, 12, 35, 43, 55, 78, 82, 90) \). Describe the sequence of integers in \( A \) that are compared to 90.

Solution. 43, 78, 82, 90.

Problem 6 (10 marks). Suppose that we use the quick sort algorithm to sort array \( A = (35, 12, 82, 55, 5, 90, 43, 78) \). Recall that the algorithm first randomly chooses a pivot, and then recursively sorts two arrays. Suppose that the pivot picked is 12. List the elements in the two arrays, respectively (ordering of the elements does not matter).

Solution. Left array: 5. Right array: 35, 12, 82, 55, 90, 43, 78.

Problem 7 (20 marks). Let \( A \) be an array of \( n \) integers. Report the pair \((i, j)\) maximizing \( A[j] - A[i] \) among all the possible \((i, j)\) satisfying \( 1 \leq i < j \leq n \). Your algorithm must terminate in \( O(n) \) time. For example, if \( A = (35, 20, 80, 60, 90, 45) \), then the answer is \((2, 5)\) with \( A[5] - A[2] = 70 \).

- If $A[i] > A[m]$, then update $m = i$.

At the end, return $(i^*, j^*)$.

Problem 8 (25 marks). Let $S_1$ be a set of $n$ integers, and $S_2$ another set of $\log_2 n$ integers ($n$ is a power of 2). Each set is given in an array (which is not sorted). Describe an algorithm to output the number of pairs $(x, y)$ satisfying $x \in S_1$, $y \in S_2$, and $x \leq y$. Your algorithm must terminate in $O(n \log \log n)$ time. For example, if $S_1 = \{10, 7, 12, 18\}$ and $S_2 = \{15, 7\}$, then you should output 4 because 4 pairs satisfy the required conditions: $(10, 15), (7, 15), (12, 15), (7, 7)$.

Solution. Sort $S_2$ in $O(n \log n \cdot \log \log n)$ time. For every element $x \in S_1$, perform binary search on $S_2$ to find the number $t_x$ of elements in $S_2$ that are larger than or equal to $x$. Return $\sum_{x \in S_1} t_x$. Every binary search costs $O(\log \log n)$ time. The total cost is therefore $O(n \log \log n)$. 
