Problem 1: (2 points)
The following algorithm to test whether a number \( q \) is prime was presented in the first lecture:

```plaintext
PRIME(q)
1 for i ← 2 to \( \sqrt{q} \) do
2 if i divides q then return NO
3 return YES
```

Show that \( \text{PRIME} \) is correct, that is, that \( \sqrt{q} \) is a correct upper bound for potential divisors of \( q \).

**Proof:** It suffices to show that if \( q \) has divisors different than 1 and \( q \) then it has at least one divisor between 2 and \( \sqrt{q} \). We prove it by contradiction: suppose that \( q = x \cdot y \) and that \( x > \sqrt{q} \) and \( y > \sqrt{q} \), then \( x \cdot y > \sqrt{q} \cdot \sqrt{q} = q \), that is, \( q > q \), contradiction.

Problem 2: (2 points)
a divide an conquer algorithm \( A \) splits the original problem in two related subproblems of size \( \sqrt{n} \) and then needs \( \log n \) time to combine the solutions of the subproblems into a solution for the original problem. What’s the running time of \( A \)?

**Proof:** The recurrence for the running time of \( A \) is (1 point)

\[
T(n) = 2T(\sqrt{n}) + \log n.
\]

Let’s solve this recurrence (1 point) now. Substituting \( n = 2^m \), we get that \( T(2^m) = 2T(2^{m/2}) + m \); notice that \( m = \log n \). Further, substituting \( S(m) = T(2^m) \), we obtain the new recurrence \( S(m) = 2S(m/2) + m \) which is the same as the one for merge sort, so its solution is \( S(m) = m \log m \). Substituting back \( S(m) = T(2^m) \) and \( m = \log n \), we obtain that \( T(n) = \log n \cdot \log \log n \).

Problem 3: (6 points)
Let \( A = \{3, 9, 5, 3, 1, 4, 8, 7\} \). Illustrate the execution of

1. \( \text{INSERTION-SORT}(A) \);
2. \( \text{SELECTION-SORT}(A) \);
3. \( \text{QUICK-SORT}(A, 1, 8) \); do not illustrate the execution of \( \text{PARTITION} \);
4. \( \text{MERGE-SORT}(A, 1, 8) \); do not illustrate the execution of \( \text{MERGE} \);
5. \( \text{HEAP-SORT}(A) \); do not illustrate the executions of \( \text{BUILD-HEAP} \) and \( \text{HEAPIFY} \);
6. \( \text{COUNTING-SORT}(A, 9) \).

**Proof:** We only show how the array \( A \) is modified. This is enough to show that you understood the sorting algorithms. There were many different solutions in your quizzes; I considered them all correct if it was clear that you understood the algorithms.
1. **Insertion-Sort**
   
   (3, 9, 5, 3, 1, 4, 8, 7)  
   (3, 9, 5, 3, 1, 4, 8, 7)  
   (3, 5, 9, 3, 1, 4, 8, 7)  
   (3, 3, 5, 9, 1, 4, 8, 7)  
   (1, 3, 3, 5, 9, 4, 8, 7)  
   (1, 3, 3, 4, 5, 9, 8, 7)  
   (1, 3, 3, 4, 5, 8, 9, 2)  
   (1, 3, 3, 4, 5, 7, 8, 9)

2. **Selection-Sort**
   
   (3, 9, 5, 3, 1, 4, 8, 7)  
   (1, 9, 5, 3, 3, 4, 8, 7)  
   (1, 5, 9, 3, 3, 4, 8, 7)  
   (1, 3, 9, 5, 3, 3, 4, 8, 7)  
   (1, 3, 5, 9, 3, 3, 4, 8, 7)  
   (1, 3, 3, 3, 9, 5, 4, 8, 7)  
   (1, 3, 3, 4, 9, 5, 4, 8, 7)  
   (1, 3, 3, 4, 5, 9, 8, 7)  
   (1, 3, 3, 4, 5, 7, 8, 9)

3. **Quick-Sort**
   
   The array $A$ is first partitioned as (consider that the pivot is $A[1]$) $\langle 1, 3, 3, 9, 5, 4, 8, 7 \rangle$. Then $\text{Quick-Sort}(A, 1, 3)$ and $\text{Quick-Sort}(A, 4, 8)$ are called. The first is not going to change the array $A$, so we only illustrate the second. The subarray $\langle 9, 5, 4, 8, 7 \rangle$ is again partitioned (the pivot is now) modifying $A$ to $\langle 1, 3, 3, 7, 5, 4, 8, 9 \rangle$ and then $\text{Quick-Sort}(A, 4, 7)$ is called. We keep doing this and obtain $\langle 1, 3, 3, 4, 5, 7, 8, 9 \rangle$.

4. **Merge-Sort**
   
   (3, 9, 5, 3, 1, 4, 8, 7)  
   (3, 9, 5, 3) (1, 4, 8, 7)  
   (3, 9) (5) (3) (1) (4) (8) (7)  
   (3, 9) (3, 5) (1, 4) (7, 8)  
   (3, 3, 5, 9) (1, 4, 7, 8)  
   (1, 3, 3, 4, 5, 7, 8, 9)

5. **Heap-Sort**
   
   The procedure $\text{Build-Heap}$ yields $A = \langle 9, 7, 8, 3, 1, 4, 5, 3 \rangle$. Then the following changes generated by swap-ings and heapifies end up with the sorted array:
   
   (3, 7, 8, 3, 1, 4, 5, 9)  
   (8, 7, 5, 3, 1, 4, 3, 9)  
   (3, 7, 5, 3, 1, 4, 8, 9)  
   (7, 3, 5, 3, 1, 4, 8, 9)  
   (4, 3, 5, 3, 1, 7, 8, 9)  
   (5, 3, 4, 3, 1, 7, 8, 9)  
   (1, 3, 4, 3, 5, 7, 8, 9)  
   (4, 3, 1, 3, 5, 7, 8, 9)  
   (3, 3, 1, 4, 5, 7, 8, 9)  
   (1, 3, 3, 4, 5, 7, 8, 9)  
   (1, 3, 3, 4, 5, 7, 8, 9)

6. **Counting-Sort**
   
   The frequency array is $F = \langle 1, 0, 2, 1, 1, 0, 1, 1, 1 \rangle$. After the next step, it becomes $F = \langle 1, 1, 3, 4, 5, 5, 6, 7, 8 \rangle$. Next, we visit the elements in $A$ from the last one toward the first and output them in $B$ according to $F$, appropriately decreasing the frequencies. We get $B = \langle 1, 3, 3, 4, 5, 7, 8, 9 \rangle$ and $F = \langle 0, 1, 1, 3, 4, 5, 5, 6, 7 \rangle$. 