You receive a sales call from a new start-up called MYPD (which stands for “Manage Your Priorities... Differently”). The MYPD agent tells you that they just developed a ground-breaking comparison-based priority queue. This queue implements Insert in time $\log_2(\sqrt{n})$ and Extract_max in time $\sqrt{\log_2 n}$. Explain to the agent that the company can soon be sued by its competitors because either (1) the queue is not comparison-based; or (2) the queue implementation is not correct; or (3) the running time they claim cannot be so good. To put differently, no such comparison-based priority queue can exist.

(Hint: Be very precise with the constant $c$ s.t. $\log_2(n!) \approx cn \log n$.)

Solution: **************** INSERT YOUR SOLUTION HERE ****************
You are given two sorted arrays $A[1, \ldots, n]$ and $B[1, \ldots, n]$, and you want to find a median of all the $2n$ numbers (i.e., the $n$-th ranked element of the "merge" of $A$ and $B$). Assume $n = 2^k$ and ignore floors and ceilings.

(a) (3 points) Assume you compare $A\left[\frac{n}{2}\right]$ and $B\left[\frac{n}{2}\right]$ of $A$ and $B$ (Please ignore floors and ceiling; say, assume $n$ is a power of 2.) and, say, $A\left[\frac{n}{2}\right] \leq B\left[\frac{n}{2}\right]$. Argue that $\text{MEDIAN}(A[1, \ldots, n], B[1, \ldots, n]) = \text{MEDIAN}(A[\frac{n}{2} + 1, \ldots, n], B[1, \ldots, \frac{n}{2}])$.

(b) (6 points) Design an $O(\log n)$ divide-and-conquer algorithm to compute the median of all $2n$ numbers. Do not forget the base case $n = 1$.

Write the exact recurrence equation for the number of comparisons made by your algorithm and solve it exactly (no $O()$-notation). (Hint: Write a recursive procedure $\text{FIND-MEDIAN}(A, a, B, b, i)$, which finds the median of $A[a, \ldots, a + i - 1] \cup B[b, \ldots, b + i - 1]$, with the initial call $\text{FIND-MEDIAN}(A, 1, B, 1, n)$.)

(c) (3 points) Notice that your algorithm is a comparison-based algorithm. This means that for every $n$, its execution can be viewed as a decision-tree, where the internal nodes are the comparisons "$A[\ldots] < B[\ldots]?"$ made, tree edges are labeled "yes" and "no", and the leaves are labeled by the tuple (array-name, index) representing whether the $n$-th largest element comes from $A$ or $B$, and what its index is. Now consider any comparison-based algorithm $Z$, which (correctly) computes the $n$-th element of $A \cup B$. How many distinct leaves (i.e., correct answers) should such a decision tree have?

Number of leaves is at least .................

Use the answer above to show that the running time of any comparison-based algorithm for the problem is at least (give precise answer, without $O$)

.................

(Hint: If you solve all the parts correctly, you will conclude that the algorithm in part (b) is optimal.)
Modify counting sort so that it is still stable but you iterate from 1 to $n$ forward (not backward) by filling the dots:

```
FORWARD-COUNTING-SORT($A, n, k$
allocate new arrays $B[1 \ldots n]$, $C[1 \ldots k + 1]$
For $i = 1$ to $k$
    $C[i] = 0$
For $j = 1$ to $n$
    $C[A[j]] = C[A[j]] + 1$
$C[k + 1] = \quad$
For $i = k$ downto 1
    \quad
For $j = 1$ to $n$
    $B[\quad] = \quad$
    \quad = \quad + 1$
```

Briefly justify why your solution is correct

Solution: **************** INSERT YOUR SOLUTION HERE ****************

**** INSERT YOUR NAME HERE ****, Homework 6, Problem 3, Page 1
(a) (4 points) Assume \( n \) is even. Given an array \( A \) of size \( n \) and the fact that there is an element \( x \) that occurs at least \( 1 + n/2 \) times in \( A \), design an \( O(n) \) time algorithm to find \( x \).

\[ \text{Solution: INSERT YOUR SOLUTION HERE} \]

The Criminal Investigation Unit, while investigating a certain crime, found a set of \( n \) fingerprints of which they are convinced that more than half (i.e. \( 1 + n/2 \)) belong to the same criminal, but they are not sure which ones. They hire a fingerprint expert who can compare any two fingerprints manually and tell whether these two are the same or not. (Note, there is no meaning to "greater" or "less" here, so a "comparison" will only return "equal" or "not equal".) A naive algorithm would have the expert compare all \( n(n-1)/2 \) pairs of fingerprints, it will take a lot of time and resources. In the next two parts you will design an \( O(n) \) algorithm instead.

(b) Imagine we split \( n \) fingerprints (arbitrarily) into \( n/2 \) pairs \( P_1, \ldots, P_{n/2} \). Each pair \( P_i \) can be labeled into one of the following 3 classes: \((C, C)\), \((S, S)\), and \((C, S)\), where \( C \) denotes the criminal and \( S \) denotes someone else. Show that there must be more \((C, C)\) pairs than the \((S, S)\) pairs.

\[ \text{Solution: INSERT YOUR SOLUTION HERE} \]

(c) Based on part (b), design an algorithm to help the fingerprint expert find a strategy to find the subset of more than half identical fingerprints, where the number of comparisons is only \( O(n) \). (Hint: Try to use the idea of part (b) to reduce the problem into the same kind of problem, but on at most \( n/2 \) fingerprints, by cleverly "dropping" at least one (and sometimes both!) fingerprint(s) in each pair from part (b).)

\[ \text{Solution: INSERT YOUR SOLUTION HERE} \]

(d) (3 points) Write the recurrence equation for the running time of your algorithm and analyze it.

\[ \text{Solution: INSERT YOUR SOLUTION HERE} \]
Illustrate the operation of Radix-Sort on the following English words: BEAR, FEAR, CLAN, MASK, CLAP, MAKE, RAKE, FAKE, BORE, HAIR, WORD, EXAM, BALD, BOLD, MARK, SLAM.

**Solution:** **************** INSERT YOUR SOLUTION HERE ****************