1. Suppose you have an array $S$ of size $n$, where each element in $S$ represents a different vote for class president, where each vote is given as an integer representing the student ID of the candidate. Without making any assumptions about who is running or how many candidates there are, design an $O(n \log n)$ algorithm to determine which candidate receives the most votes. [6 points]

2. Consider a modification to the previous problem to a situation where we know the number $k < n$ of candidates running. Design an $O(n \log k)$ algorithm to determine which candidate receives the most votes. [6 points]

3. Given the input of the first problem, give an $O(n)$ time algorithm to determine if some candidate received a majority ($\lceil \frac{n+1}{2} \rceil$) of the votes.

   • $O(n)$ average or expected time is OK
   • so the hint is to look for the median

   [6 points]

4. Suppose you have a set $A$ of $n$ nuts and a set $B$ of $n$ bolts, such that each nut in $A$ has a unique matching bolt in $B$. The only kind of comparison you can make is to take a nut-bolt pair $(a, b)$, where $a \in A$ and $b \in B$, and test to see whether the threads of $a$ are larger, smaller, or a perfect match to the threads of $b$. Give an efficient (randomized) algorithm to match up all the nut and bolts. What can you say about the run-time of your algorithm? [6 points]

Total: 24 points