An array $A[0 \ldots (n-1)]$ is called rotation-sorted if there exists some cyclic shift $0 \leq c < n$ such that $A[i] = B[(i + c) \bmod n]$ for all $0 \leq i < n$, where $B[0 \ldots (n-1)]$ is the sorted version of $A$.\(^1\) For example, $A = (2, 3, 4, 7, 1)$ is rotation-sorted, since the sorted array $B = (1, 2, 3, 4, 7)$ is the cyclic shift of $A$ with $c = 1$ (e.g., $1 = A[4] = B[(4 + 1) \bmod 5] = B[0] = 1$). For simplicity, below let us assume that $n$ is a power of two (so that we can ignore floors and ceilings), and that all elements of $A$ are distinct.

(a) (4 points) Prove that if $A$ is rotation-sorted, then one of $A[0 \ldots (n/2-1)]$ and $A[n/2 \ldots (n-1)]$ is fully sorted (and, hence, also rotation-sorted with $c = 0$), while the other is at least rotation-sorted. What determines which one of the two halves is sorted? Under what condition both halves of $A$ are sorted?

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

(b) (8 points) Assume again that $A$ is rotation-sorted, but you are not given the cyclic shift $c$. Design a divide-and-conquer algorithm to compute the minimum of $A$ (i.e., $B[0]$). Carefully prove the correctness of your algorithm, write the recurrence equation for its running time, and solve it. Is it better than the trivial $O(n)$ algorithm? (Hint: Be careful with $c = 0$ an $c = n/2$; you might need to handle them separately.)

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

---

\(^1\)Intuitively, $A$ is either completely sorted (if $c = 0$), or (if $c > 0$) $A$ starts in sorted order, but then "falls off the cliff" when going from $A[n - c - 1] = B[n - 1] = \max$ to $A[n - c] = B[0] = \min$, and then again goes in increasing order while never reaching $A[0]$. 

**** INSERT YOUR NAME HERE ****, Homework 3, Problem 1, Page 1
Let $A$ be an array with $n$ distinct integer elements in sorted order. Consider the following algorithm $\text{IdFind}(A, j, k)$ that finds an $i \in \{j \ldots k\}$ such that $A[i] = i$, or returns $\text{FALSE}$ if no such element $i$ exists.

1 $\text{IdFind}(A, j, k)$
2 \hspace{1em} \textbf{If} $j > k$ \textbf{Return} FALSE
3 \hspace{1em} Set $i := \ldots$
4 \hspace{1em} \textbf{If} $A[i] = \ldots$ \textbf{Return} \ldots
5 \hspace{1em} \textbf{If} $A[i] < \ldots$ \textbf{Return} $\text{IdFind}(A, \ldots, \ldots)$
6 \hspace{1em} \textbf{Return} $\text{IdFind}(A, \ldots, \ldots)$

(a) (3 points) Fill in the blanks (denoted $\ldots$) to complete the above algorithm.

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

(b) (5 points) Prove correctness and analyze the running time of the algorithm.
(Notice the emphasis on Prove, you can’t just say “my algorithm works because it works”.)

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

(c) (2 points) Does the algorithm work if the elements of $A$ are not distinct? Why or why not?

Solution: ****************** INSERT YOUR SOLUTION HERE ******************
Find a divide-and-conquer algorithm that finds the maximum and the minimum of an array of size \( n \) using at most \( 3n/2 \) comparisons.

(Hint: First, notice that we are not asking for some iterative algorithm (which is not hard). We are asking for you to explicitly use recursion. In fact, your divide/conquer step should take time \( O(1) \). Also, you have to be super-precise about constants and the initial case \( n = 2 \) to get the correct answer.)

Solution: ***************** INSERT YOUR SOLUTION HERE ************ ***
Let \( n \) be a multiple of \( m \). Design an algorithm that can multiply an \( n \)-bit integer with an \( m \)-bit integer in time \( O(nm^{\log_2 3 - 1}) \).

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