Before Friday's 6:00 PM lab, I encourage you to try each of these, either alone or with friends in the class. I will present solutions to these in lab, along with tips for how to solve problems like them.

1. A **cycle graph** is a graph that is just a single big cycle. Every vertex has degree 2, and there are the same number of vertices and edges.

   Consider a cycle graph of size n where every vertex has a numerical value associated with it. Design an $\mathcal{O}(\log n)$ divide-and-conquer algorithm that finds and returns the value of a **local minimum** in it. A vertex is a local minimum if its value is less than those of both its neighbors. You may assume that all values are distinct.

   The cycle graph is represented as an array of numbers. (You can imagine that the end of the array wraps around so that the last value is adjacent to the first.)

2. Suppose you have two disjoint sets A and B with each with n comparable elements. More formally:

   - $n = |A| = |B| \geq 2$
   - $A \cap B = \emptyset$

   You can only perform the following operations on these sets at the costs of their stated time complexities:

   - size(S: Set) $\rightarrow$ Integer
     Description: Returns $|S|$
     Time Complexity: $\Theta(1)$
   - middleElements(S: Set) $\rightarrow$ (Element, Element)
     Description: Returns the two median elements of the set (same element if $|S|$ is odd).
     Time Complexity: $\Theta(1)$
   - discardLeft(S: Set) $\rightarrow$ Set
     Description: Returns a new set where every element less than middleElements(S)[0] is removed
     Time Complexity: $\Theta(1)$
   - discardRight(S: Set) $\rightarrow$ Set
     Description: Returns a new set where every element greater than middleElements(S)[1] is removed
     Time Complexity: $\Theta(1)$
   - bruteForce(A: Set, B: Set) $\rightarrow$ (Element, Element)
     Description: Finds the two median elements of $A \cup B$
     Time Complexity: $\Theta(|A \cup B|)$

   Write or describe an $\mathcal{O}(\log n)$ time algorithm to find the two median elements of $A \cup B$. Do not assume anything about the datatype Element except that it is comparable ($<, =, >$ are valid operations). Briefly justify why your algorithm takes $\mathcal{O}(\log n)$ time.

3. Recently, I was looking back at my career as a professor and I was thinking of some of my former students, each of whom has already graduated and moved onto a great career, as I hope all of you will soon (I will miss you though). I was wondering which pair of students were *both* students concurrently for the longest period of time. To answer this question, I had to gather some information. From the registrar, I was able to get very precise enrollment and graduation dates for each. The registrar, for purposes of this question anyway, defines a student's **enrollment** time as the precise millisecond that the student's intent to register is recorded, and defines the student's **graduation** time as the precise millisecond that the student's diploma is printed. The time from enrollment to graduation is an individual's time as a student. Note that the definitions for enrollment and graduation in this problem mean that no two students have the exact same enrollment time, nor do any two have the exact same graduation time, even if they started the same quarter and ended the same quarter.

   Suppose you are given a list of n students, each with the enrollment and graduation times as described above. You may assume the list is sorted by enrollment date if you prefer. Design an efficient **divide and conquer** algorithm to determine which two individuals' time as students has the maximum overlap.

   More formally, you want to find the pair of indices $i \neq j$ such that the following value is maximized:

$$\min(A[i].\text{graduation}, A[j].\text{graduation}) - \max(A[i].\text{enrollment}, A[j].\text{enrollment})$$

   In a few sentences, justify the correctness of your algorithm. Also, set up and solve a recurrence to justify the running time of your algorithm.

   No credit will be given for an algorithm with running time $\Omega(n^2)$.

   Do not assume people graduate in the order they register. Do not assume everyone graduates in four years, or 12 quarters, or 45 months, or anything like that.