The last two problems of this assignment covers greedy algorithms, which we will begin to cover on next Wednesday. So you could wait till Wednesday to work on these two problems.

1. The well-known mathematician George Polya posed the following false “proof” showing through mathematical induction that actually, all horses are of the same color.

   **Base case:** If there’s only one horse, there’s only one color, so of course its the same color as itself.

   **Inductive case:** Suppose within any set of \( n \) horses, there is only one color. Now look at any set of \( n + 1 \) horses. Number them: 1, 2, 3, \ldots, \( n \), \( n + 1 \). Consider the sets \{1, 2, 3, \ldots, \( n \}\} and \{2, 3, 4, \ldots, n + 1\}\}. Each is a set of only \( n \) horses, therefore within each there is only one color. But the two sets overlap, so there must be only one color among all \( n + 1 \) horses.

   Identify what is wrong with this proof.

2. Given two sorted arrays \( a[1, \ldots, n] \) and \( b[1, \ldots, n] \), given an \( O(log n) \) algorithm to find the median of their combined \( 2n \) elements. (Hint: use divide and conquer).

3. Given an array \( A \) of \( n \) distinct numbers whose values \( A[1], A[2], \ldots, A[n] \) is unimodel: that is, for some index \( p \in [1, n] \), the values in the array first increases up to position \( p \), then decrease the remainder of the way. For example \([1, 2, 5, 9, 7, 3]\) is one such array with \( p = 4 \). Please design an \( O(log n) \) algorithm to find the peak \( p \) given such an array \( A \).

4. **Interval scheduling.** We are given a set of requests for using a resource. Each request \( i \) specifies a starting time \( s(i) \) and an end time \( f(i) \). The resource can only accommodate one request at a time. If two requests overlap in time, they are incompatible and cannot be both fulfilled. The goal is to identify a maximum subset of compatible requests. One possible greedy strategy is to select at each step the request that is compatible with the maximum number of the remaining requests. Will this greedy strategy lead to an optimal solution? If so, provide a proof. If not, provide a counter example.
5. You and your friends are taking a long hiking trip of $L$ miles, along which there are $n$ camping sites located at distances $x_1, x_2, \ldots, x_n$ respectively from the start of the trip. You can hike at most $d$ miles per day, by the end of which you must stop and camp for the night. You need make a valid trip plan that takes the minimum number of camping stops. The plan should specify which camping sites to use, and it is only valid if any two consecutive stops are no more than $d$ miles apart. Your friend proposed the following strategy: each time you come to a camp site, check whether you can make it to the next site before the end of the day (i.e., before finishing the $d$ miles quota for the day. We assume this can always be determined correctly), if so, keep hiking. If not, stop for the night.

This is in fact a greedy algorithm, which simply chooses to hike as long as possible each day. Prove that this greedy algorithm achieves the optimal solution, i.e, it uses the minimum number of stops. (Hint: construct a proof that is similar to the interval scheduling proof, which shows that the greedy algorithm stays ahead.)