Question 1
You’ve just come up with the brilliant idea of opening a chain of combination coffee shop & taco stands all along Route 90 between Syracuse and Albany. ’Cuz who doesn’t like the idea of ordering a tall mochaccino latte and two fish tacos? You’ve already done some consumer research, and for every rest area where you can build a store, you’ve got a pretty good idea of how much money you could make by doing so.

You’d open a store at each of the \( n \) rest areas, except for two concerns. First, you realize that by opening stores at neighboring rest stops, you’re basically competing with yourself. So you don’t want to do that. Second, some bozo has introduced “anti-monopolistic legislation,” which says that you can open at most \( k \) stores in total.

So more formally, you are given \( n \) locations along the highway. For each location \( i \), you have an estimate \( r_i \) for how much money you’d earn by building at that location. Your goal is to find a set \( S \) of locations at which to build such that \( |S| \leq k \), if \( i \in S \) then \( i + 1 \notin S \), and subject to these constraints, \( \sum_{i \in S} r_i \) is maximized.

Give an efficient algorithm to solve this problem.

Example  If \( n = 5 \) with \( r_1 = 8 \), \( r_2 = 10 \), \( r_3 = 7 \), \( r_4 = 4 \), and \( r_5 = 9 \), then if \( k = 2 \) then the optimal solution is to build at locations 2 and 5. If \( k \) were 3 or more, then the optimal solution would be to build at locations 1, 3 and 5.

Question 2
You are interested in analyzing some hard-to-obtain data from two separate databases. Each database contains \( n \) numerical values — so there are \( 2n \) values in total — and you may assume that no two values are the same. You’d like to determine the median of this set of \( 2n \) values, which we will define here to be the \( n^{th} \) smallest value.

However, the only way you can access these values is through queries to the database. In a single query, you can specify a value \( k \) to one of the two databases, and the chosen database will return the \( k^{th} \) smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible.

Give an algorithm that finds the median value using at most \( O(\log n) \) queries. You may assume that \( n \) is a power of two.

(Except for the last assumption, this is exercise 5.1 from the textbook.)
Question 3

Suppose that you’re consulting for a bank that’s concerned about fraud detection, and they come to you with the following problem. They have a collection of \( n \) bank cards that they’ve confiscated, suspecting them of being used in fraud. Each bank card is a small plastic object, containing a magnetic stripe with some encrypted data, and it corresponds to a unique account in the bank. Each account can have many bank cards corresponding to it, so we’ll say that two bank cards are equivalent if they correspond to the same account.

It’s very difficult to read the account number off a bank card directly, but the bank has a high-tech “equivalence tester” that takes two bank cards and, after performing some computations, determines whether they are equivalent.

Their question is the following: among the collection of \( n \) cards, is there a set of more than \( n/2 \) of them that are all equivalent to one another? Assume that the only feasible operations that you can do with the cards are to pick two of them and plug them into the equivalence tester. Show how to decide the answer to their question with only \( O(n \log n) \) invocations of the equivalence tester. You may again assume that \( n \) is a power of two.

(Except for the last assumption, this is exercise 5.3 from the textbook.)