1. Consider maintaining a collection of lists of items on which the following operations can be performed:

(i) Given two lists $L_1$ and $L_2$, form their concatenation $L$ (destroying $L_1$ and $L_2$ in the process).

(ii) Given a list $L$ and a positive integer $k$, split $L$ into two lists $L_1$ and $L_2$, where $L_1$ consists of the first $k$ items of $L$, and $L_2$ the rest ($L$ is destroyed in the process).

(iii) Report the first item in given list $L$.

(iv) Create a new list with one item.

Describe data structures and algorithms supporting these operations so that operations (i)–(iii) can be performed in time $O(\log n)$ (where $n$ is the length of $L$), and operation (iv) takes constant time.

2. Design a data structure that will hold up to $n$ data items, and supports the following operations: Insert, DeleteMin, and Lookup. The operations Insert and DeleteMin should take time $O(\log n)$ and the operation Lookup should take expected constant time. Assume that comparing two data items takes constant time. Your solution should make use of (among other things) hashing with a universal family of hash functions. Make any reasonable assumptions you like about the hash functions; in particular, you may also assume that evaluating a hash function takes constant time. In specifying your algorithm, you should indicate how many slots $m$ your hash table should use (as a function of $n$).

3. Suppose we wish to hash $n$ data items into $m$ slots using a universal family of hash functions. How big should $m$ be, relative to $n$, in order to ensure that the probability that there are no collisions at all is at least 1/2.

4. Suppose we have $n$ data items, and want to detect if there are any duplicates among them. Show how to do this in expected time $O(n)$ using hashing. To do this, you may assume that you have a universal family of hash functions that maps the data items into a number of slots of your choosing. You may also assume that the operations of evaluating a hash function and comparing if two items are equal takes constant time.

In specifying your algorithm, you should indicate how many slots $m$ your hash table should use (as a function of $n$), and then carefully prove that the overall expected running time is $O(n)$. You may use any results proved in class.

5. Honors Exercise. Extending Exercise 1, suppose we also want to an operation that reverses a given list $L$. Show how this operation can be implemented in constant time, while the other operations can still be performed within the time bounds.