CSE 332: Data Abstractions
Assignment #5
July 25, 2011
due: Monday, August 1, 10:50 a.m.

1. The `remove` procedure of Figure 5.17 for deletion from a hash table with open addressing uses "lazy deletion".

   (a) Starting from an empty hash table, give an example with *as few dictionary operations as possible* that demonstrates that using "full deletion" can cause the hash table to return the incorrect result for some operation. Make your example complete:

   - State the table size, probing strategy, and hash function.
   - Provide the sequence of operations and the state of the hash table after each operation.
   - Demonstrate how lazy deletion leads to the correct result.
   - State the incorrect result that will occur using full deletion.

   (b) When rehashing to a larger table, do lazily-deleted items need to be included? Explain your answer.

2. Exercise 6.2. Show your heaps using trees rather than arrays. Show the heap after every insertion in part (a) and after every `percolateDown` that changes the heap in part (b).

3. Exercise 6.3. Just apply these operations starting with the tree from Exercise 6.2(b). Show the tree after each `deleteMin`.

4. Give an algorithm (in pseudocode or Java) that outputs all keys less than x in a binary heap, without changing the heap. The keys need not be output in sorted order. Your algorithm should run in time $O(L)$, where L is the number of keys that are output; note that this generalizes the fact that `findMin` runs in time $O(1)$. (Hint: recursion will help.)