CSE 332: Data Abstractions
Assignment #7
November 21, 2014
due: Wednesday, December 3, 12:30 p.m., *before lecture begins*

Bundles: The problems in each written homework assignment will be divided into 2 groups (to facilitate distribution to grading TAs). You will turn in 2 corresponding bundles. Write your full name in the *upper left corner* of each bundle's top page, with your last name printed clearly in CAPITAL LETTERS. Each bundle should be stapled separately. We don't supply the stapler.

This week's turnin bundles: (A) problems **1**–**3**, (B) problems **4**–**5**.

1. 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).

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

3. Exercise 9.5. Part (a) is asking for least cost paths, and part (b) is asking for shortest paths ignoring the edge costs. Show your work for part (a) as in Figures 9.21-9.27 and for part (b) as in Figure 9.19.

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.) Include an explanation of *why* your algorithm runs in time $O(L)$.

5. Given a (very long) string T called the "text" and a (short) string P called the "pattern", the *string-matching problem* is to find substrings of T that are equal to P. Let n be the length of T. You can assume that the length of P is a constant. All of your algorithms below should have work $O(n)$ and span $O(\log n)$. Be sure to explain for each algorithm why this is true.

   (a) Describe a fork-join parallel algorithm that outputs the index of the leftmost occurrence of the pattern P in T, using a sequential cutoff of 1.

   (b) Describe a fork-join parallel algorithm that outputs an array of the indices of all occurrences of the pattern P in T, using a sequential cutoff of 1. (Hint: use Pack.)

   (c) What changes do you need to make to your solution in part (a) to use a more sensible sequential cutoff?