CS 61B Sorting and Hashing Fall 2016

1 Heaps of fun®

(a) Assume that we have a binary min-heap (smallest value on top) data structure called Heap that stores integers and has properly implemented insert and removeMin methods. Draw the heap and its corresponding array representation after each of the operations below:

```
Heap h = new Heap(5); //Creates a min-heap with 5 as the root
h.insert(7);
h.insert(3);
h.insert(1);
h.insert(2);
h.removeMin();
h.removeMin();
```

(b) Consider an array based min-heap with N elements. What is the worst case running time of each of the following operations if we ignore resizing? What is the worst case running time if we take into account resizing? What are the advantages of using an array based heap vs. using a node-based heap?

```
Insert      _______
Find Min    _______
Remove Min  _______
```

(c) Your friend Alyssa P. Hacker challenges you to quickly implement an integer max-heap data structure - "Hah! I'll just use my min-heap implementation as a template to write max-heap.java", you think to yourself. Unfortunately, your arch-nemesis Malicious Mallory deletes your min-heap.java file. You notice that you still have the min-heap.class file; could you use it to complete the challenge? – you can still use methods from min-heap but you cannot modify them. If so, describe your approach, do not write code. If not, explain why it is impossible.

2 HashMap Modification (from 61BL SU2010 MT2)

(a) When you modify a key that has been inserted into a HashMap will you be able to retrieve that entry again? Explain.

☐ Always ☐ Sometimes ☐ Never

(b) When you modify a value that has been inserted into a HashMap will you be able to retrieve that entry again? Explain.

☐ Always ☐ Sometimes ☐ Never

3 Hash Codes

(a) Suppose that we represent Tic-Tac-Toe boards as 3 by 3 arrays of integers (each of which is in the range 0 to 2). Describe a good hash function for Tic-Tac-Toe boards that are represented in this manner. Try to come up with one such that boards that are not equal will never have the same hash code.

(b) Is it possible to add arbitrarily many Strings to a Java HashSet with no collisions? If not, what is the minimum number of distinct Strings you need to add to a HashSet to guarantee a collision?

4 Bonus Question

Describe a way to implement a linked list of Strings so that removing a String from the list takes constant time. You may assume that the list will never contain duplicates.