University of California at Berkeley
Department of Electrical Engineering and Computer Sciences
Computer Science Division

Spring 2012 **Jonathan Shewchuk**

CS 61B: Midterm Exam II

This is an open book, open notes exam. Electronic devices are forbidden on your person, including cell phones, iPods, headphones, and laptops. Turn your cell phone off and leave it and all electronics at the front of the room, or risk getting a zero on the exam. **Do not open your exam until you are told to do so!**

Name:_____

Login:_____

Lab TA:_____

Lab day and time:_____

Do not write in these boxes.

Problem #	**Possible**	**Score**
1. Miscellaneous	6	
2. Trees	10	
3. Graphs	9	
Total	25	

Problem 1. (6 points) **A miscellany.**

a. (1 point) Suppose that the following code compiles with no errors, and `BException` is a checked exception. What is the relationship between `AException` and `BException`?

```
public class SomeClass {
  public void method1() throws AException {
    method2();
  }

  public void method2() throws BException {
    throw new BException();
  }
}
```

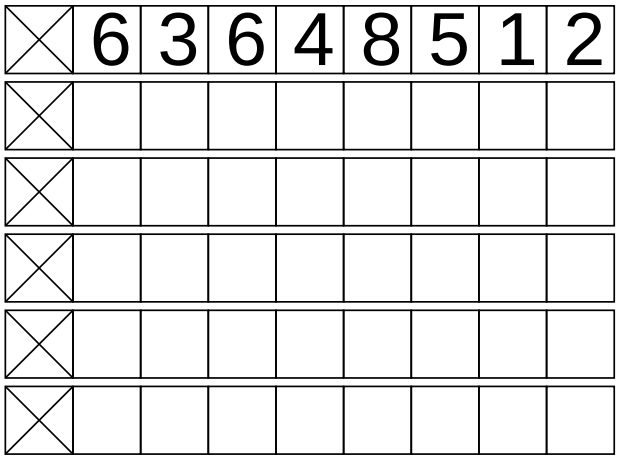
b. (1 point) Consider a hash table that initially has 50 buckets and no entries. Whenever the load factor exceeds one, the array is resized to be 50 buckets larger. If no chain ever contains more than $\mathcal{O}(1)$ entries, the total time to insert n entries into the hash table, **including resizing time**, is in $\Theta($_____$)$.

c. (4 points) Prove formally and rigorously that $\log(x + y) \in \mathcal{O}(\log x + \log y)$. (Both x and y are positive variables that can grow arbitrarily large. You may assume without proof that when x gets bigger, $\log x$ gets bigger.) Hint: What if $x \geq y$? What if $y \geq x$?

Problem 2. (10 points) **Trees.**

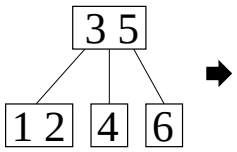
a. (2 points) Draw the **binary search tree** whose preorder traversal is `3 0 2 1 6 4 5 7`.

b. (3 points) Show the sequence of swaps by which the method `bottomUpHeap` converts the following array into a **binary heap**. Redraw the array once for **each swap** in the boxes provided.



c. (1.5 points) A **binary heap** of height h contains at least _____ keys. A **binary search tree** of height h contains at least _____ keys. A **2-3-4 tree** of height h contains at least _____ keys. (Give exact values, not asymptotic.)

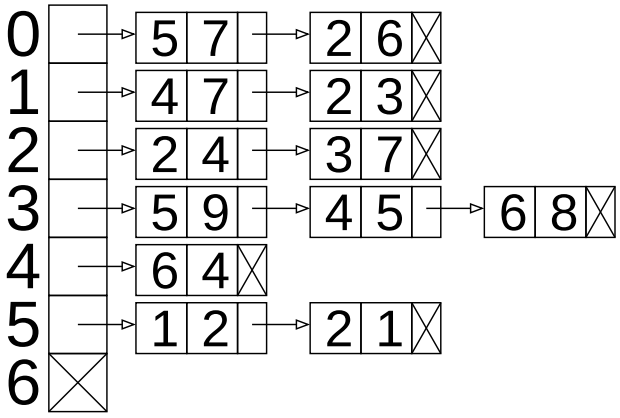
d. (1.5 points) Draw the following **2-3-4 tree** after you execute `remove(6)`.



e. (2 points) Suppose you are given an array `k` of n integer keys, no two of them equal. Describe in plain English or Java-like pseudocode—your choice—an algorithm to answer the following question: if the n keys `k[0]`...`k[`$n-1$`]` are inserted into a **binary search tree** in order, what will be the depth of the minimum key? Your algorithm **must** run in $\mathcal{O}(n)$ time and use only $\mathcal{O}(1)$ memory (not counting the memory that stores the array `k`), so constructing a binary search tree is **not** an option.

Problem 3. (9 points) **Graphs.**

a. (2 points) In the adjacency list pictured below, each listnode contains three fields: the destination of an edge, the weight of an edge, and a reference to the next listnode, in that order. For example, the first listnode in the first linked list represents the directed edge (0, 5) with weight 7. Suppose we run the breadth-first search algorithm from Lecture 29 on the **directed graph** this adjacency list represents, starting at vertex 0. Recall that when BFS visits a vertex, it creates a `parent` field pointing from the visited vertex to the vertex that BFS reached it from. Draw the **directed tree** that these `parent` references represent. **Label its vertices** A, B, C, D, E, F, G in the order they're visited by BFS.



b. (1 point) We can make this graph strongly connected by adding just one edge, namely (___, ___).

c. (2 points) Draw **all** the minimum spanning trees of this graph, treating it as an undirected graph.

d. (2 points) Suppose G is a connected, weighted, undirected graph that has many vertices and many edges—no two with the same weight—but G has only one minimum spanning tree T.

Do you think that T **must** include the edge of G with the least weight? ____ What about the edge with the second-least weight? ____ What about the edge with the third-least weight? ____ Explain why.

e. (1 point) With an **adjacency matrix**, the worst-case running time of Kruskal's algorithm, expressed in terms of the number n of vertices and the number e of edges, is in $\Theta($_____$)$.

f. (1 point) The worst-case running time of Kruskal's algorithm on graphs with exactly 100 vertices, expressed in terms of the number e of edges, is in $\Theta($___don't answer___$)$. This is a nasty trick question that doesn't even make sense. Explain what is wrong with the question.