Final Exam Review

CIS 211

Final Exam Topics

• BST insertion, traversals
• Collections tracing
• Recursive tracing
• Programming with Stacks and Queues
• Programming with Maps and Sets
• Binary tree programming (easier)
• Binary tree programming (harder)

BST Insertion

Show the binary search tree that would result from inserting the following items in this order: "twas", "brillig", "and", "the", "slithy", "toves". Assume an alphabetical sort order for the binary search tree ('a' < 'b' < . . . < 'z'). (Note that "the" < "toves" < "twas".)

BST Traversals

• Show the node orders resulting from each traversal in the following binary tree:

  - Pre-order:
  - In order:
  - Post-order:
Recursive Tracing

What does mystery(x) print for each input?

```java
private static int helper(int x)
{
    if (x > 0) {
        return helper(x / 2) + 1;
    } else {
        return 0;
    }
}

public static void mystery(int x)
{
    System.out.println(helper(x));
}
```

5 1
5 -5
5 16

Stacks and Queues (and more)

Write a method void removeDuplicates(Queue<String> q) that removes all duplicates entries from a queue. When removing duplicate elements, the first occurrence of any given element should remain and the later ones should be removed. Elements must retain their respective orders.

Example: Suppose a queue contains the following elements before removeDuplicates:
front [1, 3, 2, 8, 2, 4, 3, 3, 1, 17] back
After calling removeDuplicates, it should contain:
front [1, 3, 2, 8, 4, 17] back

You may create AT MOST TWO auxiliary data structures (such as a Queue, Stack, TreeMap, or LinkedList). Your solution must run in O(n) time where n is the initial length of the queue.

Maps and Sets

Write a method String mostCommonGrade(Map<String, String> grades) that accepts one argument, a Map from student names to grades, and returns the most common grade in the Map. In the case of a tie, any of the most common grades may be returned.

Example: Suppose the Map grades contains the following entries:
{"Rocky"="Excellent", "Bullwinkle"="Good", "Boris"="Fair", "Natasha"="Good"}

mostCommonGrade(grades) should return "Good", since it occurs the most times.

Binary Tree Programming 1

Add the method isHeapOrdered to the IntTree class discussed in lecture. isHeapOrdered returns true if and only if the value at each node is smaller than all of its descendants.

Your solution must not create any new objects of any type. (In other words, you may not use the "new" keyword.) You may use additional private methods that you write yourself, but should call no other IntTree methods.

Examples: The left tree is heap ordered. The right one is not, since 3 is below 4 and 7 is below 9.
Binary Tree Programming 2

Add the method `pruneToDepth(int d)` to the `IntTree` class discussed in lecture. `pruneToLevel` removes all nodes below depth `d`, while leaving all other nodes where they were. If `d` is negative, `pruneToDepth` throws an `IllegalArgumentException`.

Your solution must not create any new objects of any type. (In other words, you may not use the “new” keyword.) You may use additional private methods that you write yourself, but should call no other `IntTree` methods.

For example, `pruneToDepth(0)` should result in an empty tree, `pruneToDepth(1)` should result in a tree of at most one node, and `pruneToDepth(2)` should result in a tree of at most three nodes.