

CS 2 Exam #1 Review

Outline of material covered so far:

I. Algorithm Analysis Preliminaries

- a. Running Time
- b. Mathematical Review
 - i. Logs & Exponents
 - ii. Summations
- c. Justification Techniques
 - i. By example
 - ii. By contradiction
- d. Big-Oh notation
 - i. O
 - ii. Ω
 - iii. θ
- e. Experimental Analysis
- f. Example Algorithms
 - i. Sorted List Matching Problem
 - ii. Maximum Contiguous Subsequence Sum Problem

II. Basic Data Structures

- a. Linked Lists
 - i. standard
 - ii. doubly linked
 - iii. circular
- b. Stacks
 - i. Array implementation
 - ii. Linked List implementation
- c. Queues
 - i. Array implementation
 - ii. Linked List implementation
- d. Double-Ended Queues

III. Binary Search Trees

- a. Traversals**
 - i. Preorder**
 - ii. Inorder**
 - iii. Postorder**
- b. Insert**
- c. Delete**

Format of exam:

Most of the exam will involve writing code. All of this will be writing methods given certain specifications. The rest of the questions will be short answer or true/false. Some of the coding will be based on the BinTree and BinTreeNode classes discussed in class. You will receive printouts of these for your exam. You may be asked to explain some of the code or write extra methods for either class. Any other code you need to view to answer questions will also be provided.

How to study:

- 1) Look over the notes, paying attention to all the code shown in class. Make sure you understand how it all works.**
- 2) Skim over the book, making sure you understand the general concepts discussed.**
- 3) Look over your two programming assignments, making sure you remember how you solved certain problems.**
- 4) If you have time, sketch the outline for coding up classes that we discussed, but did not code.**

Practice Questions (from previous exams)

1. A partial implementation of a double ended queue class is included below. Assume that this class supports a deq with a maximum size of 10 only. Write the code for the `removeFirst()` and `removeLast()` methods that return integers. Also assume that neither remove method is ever called on an empty deq.

```
public class deq {

    private int[] items;
    private int high;

    public deq() {
        items = new int[10];
        high = -1;
    }

    public void insertFirst(int n) {

        high++;
        for (int i=high; i>0; i--)
            items[i] = items[i-1];
        items[0] = n;
    }

    public void insertLast(int n) {
        high++;
        items[high] = n;
    }
}
```

```
public int removeFirst() {
```

```
}
```

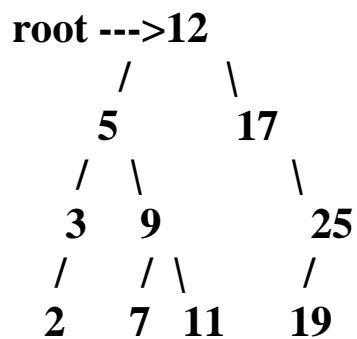
```
public int removeLast() {
```

```
}  
}
```

2. Prove, using the formal definition of O , that $3n^2 - 6n + 2 = O(n^2)$.

3.

a) (5 pts) Included in the code handout is the method `question3`. What is the return value of `question3` being called on the `BinTree` object represented below:



**b) (5 pts) What value does the method return in general?
(Your answer should be a single English sentence. Something like, "The method returns the total number of nodes stored in the binary tree with it's root at the BinTree object the method is called on.")**

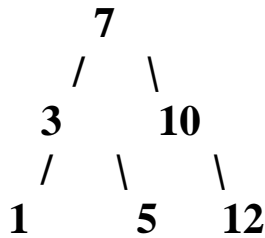
Code in BinTreeNode class:

```
public int question3() {
    if (left == null && right == null)
        return 2;
    else if (left == null)
        return 1+right.question3();
    else if (right == null)
        return 1+left.question3();
    else
        return
left.question3()+right.question3();
}
```

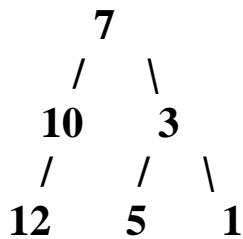
In BinTree class:

```
public int question3() {
    if (root == null)
        return 1;
    else
        return root.question3();
}
```

4. Add a method `mirrorImage()` to the `BinTreeNode` class. This method should take the tree referenced by the current object and change the tree so that it looks like a mirror image of the tree passed to it. For example, if a binary tree looked like the following:



and then `mirrorImage` was called on the `BinTreeNode` storing the 7, the resulting binary tree would look as follows:



The corresponding method in the `BinTree` class that will call yours is included in your code handout. Note: Normally, this would NOT be a method we would want to add to this class because the resulting tree does NOT satisfy the binary search tree invariant. (Hint: This method is much easier to write recursively.)

```
public void mirrorImage() {
```

```
}
```