COP 3530
Data Structures

Midsemester Exam

Name: _____

February 19, 2009

This exam has 4 questions. Each question starts on a new page. Please answer each question on its page. You may assume `java.util` has been imported. There will be no deductions for lack of commenting. There will be no deductions for lack of import directives. There will be no deductions for minor syntax errors.

1. **[50 points]** `equals`, shown below, returns true if the two lists have the same size and contains the same elements in the same order. Assume N is the size of the longer list.

```
public boolean equals( List<Integer> lhs, List<Integer> rhs )
{
    if( lhs.size( ) != rhs.size( ) )
        return false;

    for( int i = 0; i < lhs.size( ); i++ )
        if( !lhs.get( i ).equals( rhs.get( i ) )
            return false;

    return true;
}
```

   (a) What is the running time of `equals` when both lists are `ArrayList`s?

   (b) What is the running time of `equals` when both lists are `LinkedList`s?

   (c) Suppose it takes 4 seconds to run `equals` on two equally-valued 10000-item `ArrayList`s. How long will it take to run `equals` on two equally-valued 50000-item `ArrayList`s?

   (d) Explain in one sentence how to make the algorithm efficient for all types of lists.

2. **[50 points]** This question requires that you implement some methods for a class that represents a doubly-linked list. In this question, **both a header and a tail are used**. You may assume an appropriate declared nested class `Node`. You may assume that the list does not store `null` values. You should only be following links; your solutions should not create or use any iterator classes.

   (a) Below you will implement `getNode`, `remove`, and two-parameter `add`. Before writing the code, give the Big-Oh running time for each routine.

   (b) Implement `getNode`. You DO NOT have to optimize this method for the case where the index represents a node in the second half of the list. You may not invoke other methods of this class.

```
// If 0 <= idx < size( ) return a link to corresponding node
// If idx == size( ) return tail node
// Otherwise, throw an exception
// You DO NOT have to optimize for the case where idx > size()/2
private Node<AnyType> getNode( int idx )
{









}
```

   (c) Implement `remove` below. You may not invoke any other methods of the class EXCEPT that you may use `getNode` regardless of whether you correctly implemented part (a).

```
public void remove( int idx )
{









}
```

   (d) Implement `add`. You may invoke `getNode`. The new item `x` is placed in position `idx`.

```
public void add int idx, AnyType x )
{









}
```

   DID YOU REMEMBER TO GIVE THE BIG-OH?

3. [**50 points**] In the class graderoll, a seven digit Panther ID is reported with three leading Xs and the last four digits. For instance Panther ID 9544756 is reported as XXX4756. Write a method that takes as input an array consisting of Panther IDs (they may be considered to be `String`s, and returns a `List<String>` containing all codes that correspond to two or more Panther IDs. For instance, if the input array is

   ```
   [ 1234567, 2345678, 2224567, 4341240, 6545678 ]
   ```

   the returned list consists of

   ```
   [ XXX5678, XXX4567 ]
   ```

   Write this routine below, using Java 5.

4. [**50 points**] Implement a method, `findMax`, that returns the largest file in a directory. You must include all subdirectories in your search.

   The `File` class has the following useful methods:

```
boolean isDirectory( );
long length( );
File [ ] listFiles( );
String getAbsolutePath( );
```

   Implement `findMax` below.

```
// If d is a regular file, return it.
// If d is a directory, search the directory and all subdirectories
// and return the File with the largest size.
private static File findMax( File d )
{
```