COP 3530
Data Structures

Midsemester Exam

Name: _____

May 24, 2010

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]** `reverseList`, shown below, places items in the first list into the second list in reverse order.

```
public void reverseList( List<Integer> original, List<Integer> reverse )
{
    reverse.clear( );

    for( int i = 0; i < original.size( ); i++ )
        reverse.add( 0, original.get( i ) );
}
```

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

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

   (c) What is the running time of `reverseList` when `original` is an `ArrayList` and `reverse` is a `LinkedList`?

   (d) Suppose it takes 10 seconds to run `reverseList` on two 1000-item `ArrayList`s. How long will it take to run `reverseList` on two 3000-item `ArrayList`s?

   (e) Explain, in one or two sentences, how to make the algorithm efficient by using an appropriate iterator and an appropriate `add` method.

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**. However, there is no `size` field. 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 `size`, `addBefore`, and `removeNode`. Before writing the code, give the Big-Oh running time for each routine.

   (b) Implement `size`. Note that the size information is not directly stored; you have to compute it.

```
public int size( )
{









}
```

   (c) Implement `removeNode` below.

```
public void removeNode( Node<AnyType> p )
{









}
```

   (d) Implement `addBefore`. This method adds `x` prior to `Node p`.

```
public void addBefore( Node<AnyType> p, AnyType x )
{









}
```

DID YOU REMEMBER TO GIVE THE BIG-OH?

3. [**50 points**] Assume that you have a `java.util.Map` representing a phonebook in which the keys are `String`s (representing names) and the values are `List<String>`s, representing all phone numbers associated with that name.

   Write a routine, `getMostNumbers`, that returns a `List` of `String`s that are customers with the most phone numbers (this list will have size more than one if there are several customers tied for having the most phone numbers). For instance, if the map contains the three key/value pairs shown here:

   ```
   { John=[3053481000,3055551212], Jane=[2125551212], Jill=[7865551291,3053480000] }
   ```

   then the `List` returned by `getMostNumbers` is

   ```
   [John,Jill]
   ```

   (a) Write this routine below, using Java 5.

   (b) Assuming that the `Map` is a `TreeMap`, provide the Big-Oh running time of your routine.

4. **[50 points]** In this question, you will use the Java `File` class to determine the number of directories (i.e. Windows folders) that are contained in a given directory.

   Specifically, implement method `countDirectories` that returns the number of directories in a directory. This number is always at least 1 (itself) for any directory. You must recursively include all subdirectories in your search.

   The `File` class has the following useful methods:

   ```
   boolean isDirectory( );
   File [ ] listFiles( );
   ```

   Notice that `listFiles` returns an array of `File`.

   Implement `countFiles` below.

   ```
   // If d is a regular file, return 0.
   // If d is a directory, return 1 plus the number of directories in
   // each of the directory entries.
   public static int countDirectories( File d )
   {
   ```