COP 3530
Data Structures

Midsemester Exam

Name: _____

June 17, 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]** `containsAll`, shown below, returns true if the first list contains all the elements in the second list. Assume the lists are approximately the same size and have about N items each.

```
public boolean containsAll( List<Integer> bigger, List<Integer> items )
{
  outer:
    for( int i = 0; i < bigger.size( ); i++ )
    {
        Integer itemToFind = bigger.get( i );

        for( int j = 0; j < items.size( ); j++ )
            if( items.get( j ).equals( itemToFind ) ) // match
                continue outer;

        // If we get here, no entry in items matches bigger.get(i)
        return false;
    }
    return true;
}
```

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

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

   (c) Suppose it takes 10 seconds to run `containsAll` on two equally-valued 1000-item `ArrayList`s. How long will it take to run `equals` on two equally-valued 2000-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, **neither a header nor 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 `toArray`, `removeLast`, and `addFirst`. Before writing the code, give the Big-Oh running time for each routine.

   (b) Implement `toArray`. You may assume that the array is the correct size, and you do not have to throw an exception if this is not the case.

```
public void toArray( AnyType [ ] arr )
{









}
```

   (c) Implement `removeLast` below. If the list is empty, throw an exception. Be sure to correctly handle the case of removing the only element.

```
public void removeLast( )
{









}
```

   (d) Implement `addFirst`. Be sure to handle the special case of an empty list.

```
public void addFirst( AnyType x )
{




}
```

DID YOU REMEMBER TO GIVE THE BIG-OH?

3. **[50 points]** Assume that you have a `java.util.Map` in which the keys are `Integer`s and the values are `List<Integer>`s, representing a prime factorization.

   Write a routine, `getPrimes`, that returns a `List` of `Integer`s that are keys whose corresponding values are lists of size one (these keys are the prime numbers). For instance, if the map contains the four key/value pairs shown here:

   ```
   { 6=[2,3], 23=[23], 101=[101], 8000=[10,20,40] }
   ```

   then the `List` returned by `getPrimes` is

   ```
   [23,101]
   ```

   (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]** Implement a method, `countFiles`, that returns the number of files in a directory. You must include all subdirectories in your search. A directory entry itself counts as a file, thus calling `countFiles` on an empty directory returns 1, and calling it on a directory that contains only a single regular file would return 2.

   The `File` class has the following useful methods:

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

   Notice that `listFiles` returns an array of `File`, so you do not have to be bothered with forming complete path names.

   Implement `countFiles` below.

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