COP 3530
Data Structures

Midsemester Exam Version A

Name: ____________________________________

March 3, 2005

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**] Static method `removeEveryOtherItem` removes items in even positions (0, 2, 4, etc.) from a `List`. One possible implementation of `removeEveryOtherItem` is shown below:

```
public static <AnyType> void removeEveryOtherItem( List<AnyType> list )
{
    for( int i = 0; i < list.size( ); i++ )
        list.remove( i );
}
```

   (a) Provide the Big-Oh running time, with a one-line explanation, if `list` is an `ArrayList`.

   (b) Provide the Big-Oh running time, with a one-line explanation, if `list` is a `LinkedList`.

   (c) If `removeEveryOtherItem` takes 8 milliseconds for an `ArrayList` of 1000 items, approximately how long would it take for an `ArrayList` of 3000 items?

   (d) Rewrite `removeEveryOtherItem`, using an iterator, so that it is efficient and does not use any extra space besides the iterator.

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 beginMarker nor an endMarker are used**; assume the first node is accessed via `first` and the last node is accessed via `last`. 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 shuld not create or use any iterator classes.

   (a) Implement `contains` and **PROVIDE ITS BIG-OH RUNNING TIME**.

   ```
   public boolean contains( Object x )
   {









   }
   ```

   (b) Implement the private helper method `remove` in the space shown below. You must provide extra code to handle the special cases where `p` is the first or last node in the list.

   ```
   private void remove( Node p )
   {









   }
   ```

3. [**50 points**] Assume that you have a `java.util.Map` in which the keys are `String`s and the values are `List<Integer>`s. The map represents words and the line numbers on which they occur.

   Write a routine, `linesToWords`, that returns a `Map` in which the keys are line numbers, and the values are lists of `String`s representing the words on the corresponding line numbers.

   For instance, if the map contains the four key/value pairs shown here:

```
{ hello=[2,3], good=[1,2], this=[1,5], if=[1,2,3] }
```

   then the map returned by `linesToWords` is

```
{ 1=["good","this","if"], 2=["hello","good","if"], 3=["hello","if"],5=["this"] }
```

   Write this routine below, using Java 1.5.

4. [**50 pts**] Method `hasSum` returns true if the array contains zero or more items that sum to parameter `sum` exactly and false otherwise. Example: `arr` is [3, 4, 9, 1]. If called with `sum` = 8, the result is true (3,4,1); if called with `sum` = 0, the result is true (use zero elements); if called with `sum` = 11, the result is false.

   The easiest way to implement `hasSum` is to use recursion. Write a public driver and a private recursive routine with appropriate parameters. You may assume that all numbers in the problem statement have type `int`.