# COP 3530
# Data Structures

Midsemester Exam

Name: ________________________________

June 22, 2006

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**] Consider the following method, whose implementation is shown:

```
// Precondition: m represents matrix with N rows, N columns,
//               in each row, elements are increasing
//               in each column, elements are increasing
// Postcondition: returns true if some element in m stores val;
//                returns false otherwise
public static boolean contains( int [ ] [ ] m, int val )
{
    int N = m.length;

    for( int r = 0; r < N; r++ )
        for( int c = 0; c < N; c++ )
            if( m[ r ][ c ] == val )
                return true;
    return false;
}
```

An example of a matrix that satisifies the stated precondition is:

```
int [ ] [ ] m1 = { { 4, 6, 8 },
                   { 5, 9, 11 },
                   { 7, 11, 14 } };
```

(a) What is the running time of `contains`, as written above?

(b) Suppose it takes 4 seconds to run `contains` on a 100-by-100 matrix. How long will it take to run `contains` on a 400-by-400 matrix?

(c) Suppose `contains` is rewritten so that the algorithm performs a binary search on each row, returning true if any of the row-searches succeed, and false otherwise. What is the running time of this revised version of `contains`?

(d) [**EXTRA CREDIT: 15 pts**] Give an algorithm, and provide a running time bound, for `contains` that is faster (in Big-Oh terms) than both algorithms above.

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**. You may assume an appropriate declared nested class `Node`. You may assume that the list does not store `null` values. You may assume that the first node in the list is accessed by `first` and the last node is accessed by `last`, and if the list is empty, then both `first` and `last` are null. You should only be following links; your solutions should not create or use any iterator classes.

   (a) Below you will implement `toString`, `contains`, and `addLast`. Before writing the code, give the Big-Oh running time for each routine.

   (b) Implement `toString`. You may not invoke other methods of this class.

```
public String toString( )
{









}
```

   (c) Implement`contains` below. You may not invoke any other methods of the class.

```
public boolean contains( AnyType x )
{









}
```

   (d) Implement `addLast`. Make sure you have handled the special case of an empty list.

```
public void addLast( 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 `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 5.

4. [**50 points**] The method `printReverse` takes a `BufferedReader` as a parameter, prints each line in the buffered input stream `in`, and closes `in`. However, **the lines are to be output in reverse order of their occurrence**. In other words, the last line is output first, and the first line is output last. For the purposes of this exam, you may assume that there are no unusual exceptions, so any calls to `close` do not have to be in a `finally` block. The signature of the method is:

```
public static void printReverse( BufferedReader in ) throws IOException
```

Implement `printReverse` **without using any Collections API or user-written containers**. Do so by using recursion (in which you output the first line AFTER recursively outputting the subsequent lines in reverse).