COP 3530
Data Structures

Midsemester Exam

Name: ____________________________________

Email: ____________________________________

October 16, 2001

This exam has 3 questions. Each question starts on a new page. Please answer each question on its page. You may write on the back of a page.

1. [**60 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. **[70 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
```

   (a) Implement `printReverse` by using a Collections API `List` to store the lines. After you have the lines read into the `List`, you should be able to output them in reverse order.

   (b) Implement `printReverse` without using any Collections API or user-written containers. Do so by using recursion.

3. **[70 points]**

(a) Complete the implementation of the following class. Since this is a timed exam, you **do not have to error check**. Note that the data member is a `TreeMap`: Because duplicates are allowed, you will store all the unique items, along with the number of times each item occurs. You may find the `TreeMap`'s `remove`, `put`, `firstKey` and `lastKey` methods useful. (Note: `DoubleEnded` uses the default comparator to determine ordering.)

(b) **Give the running time of each operation.**

```
public class DoubleEnded
{
    public boolean isEmpty( )
      {                                                             }
    public void makeEmpty( )
      {                                                             }
    public void insert( Object x )
    {






    }
    public Object findMin( )
      {                                                             }
    public Object findMax( )
      {                                                             }
    public Object deleteMin( )
    {






    }
    public Object deleteMax( )
      { /* You don't have to write this; it is similar to deleteMin. */ }

    private TreeMap items = new TreeMap( );
}
```