CMSC132
Spring 2006
Midterm #1

Grader Use Only:

#1		(25)
#2		(25)
#3		(25)
#4		(13)
#5		(12)
Total		(100)
Honors		(10)

First Name: _____

Last Name: _____

Student ID: _____

Section time _____ **TA:** _____

I pledge on my honor that I have not given or received any unauthorized assistance on this examination.

Your signature: _____

General Rules (Read):

- This exam is closed book and closed notes.
- If you have a question, please raise your hand.
- Total point value is 100 points.
- The short answer questions are not essay questions. Strive to answer them in 1 or 2 sentences. Longer answers are not necessary and are discouraged.
- **WRITE NEATLY**. If we cannot understand your answer, we will not grade it (i.e., 0 credit).
- **PUNT RULE**: For any question, you may write PUNT, and you will get ¼ of the points for the question (rounded down). If you feel totally lost on a question, you are encouraged to punt rather than waste time writing down a bunch of vaguely related verbiage in hopes of getting some partial credit.
- **Honors section questions only count for credit for students in the honors section**.

Problem 1 Software Development (25 pts)

a. (5 pts) What is the primary difference between the waterfall model of software development and the unified model.

b. (5 pts) Explain why good code coverage is important.

c. (5 pts) Give an example of a specific bug that would not be addressed or helped by having good code coverage.

d. (5 pts) What does Extreme Programming say about testing and why is it an important part of Extreme Programming?

e. (5 pts) What are the advantages using the Model View Controller design pattern?

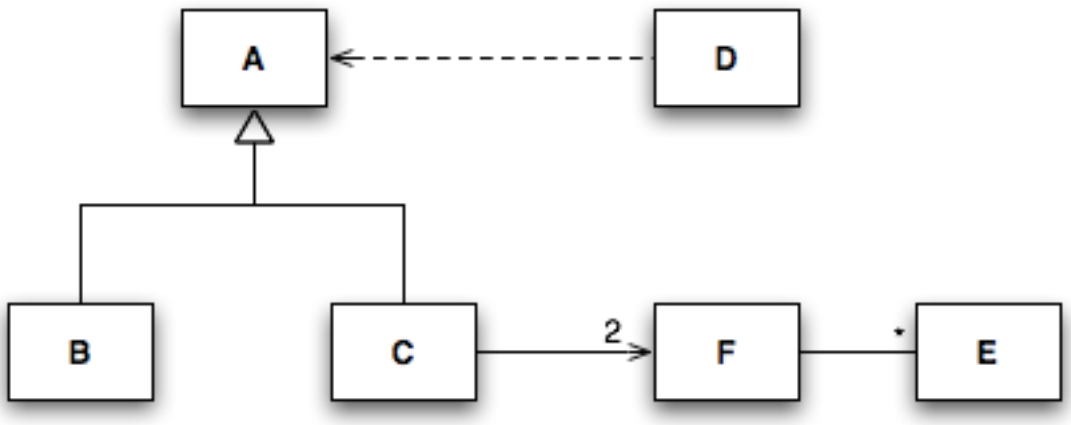
Problem 2 UML (25 pts)

a. (13 pts) Given the following Java code, draw its UML class on the next page.

```
public class Address {
        public String name, street, city, state, zip;
}

public class Postmark {
        public String date, time;
}

public class MailPiece {
        public Address sender, recipient;
        public double weight;
        public String date, time;

        public void setPostmark(String date, String time) {
                this.date = date;
                this.time = time;
        }

        public Postmark getPostmark() {
                return new Postmark(date, time);
        };
}

public class CertifiedMail extends MailPiece {
        public String receivedBy, receivedDate;
}
```

UML DIAGRAM GOES HERE

b. (12 pts) Consider the UML diagram below:



Give minimal Java class declarations for classes A-F, that you can infer from this diagram. Don't give any methods, just the class declaration header and declare what fields are indicated by this diagram.

Problem 3 Java Programming (25 pts)

a. (3 pts) What is the difference between a checked and an unchecked exception?

b. (3 pts) Rewrite the following for loop using the new enhanced for loop.

```
String[] names = {"Pete", "Rose", "Mary"};
for (int i=0; i < names.length; i++)
     System.out.println(i);
```

c. (3 pts) What would be a typical use of the `try { … } finally { … }` construct? What would go into the `try` and `finally` clauses?

d. (8 pts) Give an implementation of the iterator() method of IntegerRange that uses an anonymous inner class:

```
public class IntegerRange implements Iterable<Integer> {
     final int min,max;
     public IntegerRange(int min, int max) {
          this.min = min;
          this.max = max;
     }
     public Iterator<Integer> iterator() { … }
}
```

Calling remove on an IntegerRange iterator should throw an UnsupportedOperationException. A class that implements Iterator<Integer> needs to implement the following public methods:

```
boolean hasNext();
Integer next();
void remove();
```

Give the body of the iterator() method below:

e. (8 pts) Write a method that has the following header:

```
public static int getFirstIndexNotComplement(BitSet b1, BitSet b2)
```

Remember that a BitSet represents a set of non-negative integers, and is implemented using an sequences of bits. The method will return the index of the first bit in b1 that is not a complement of the corresponding bit in b2 (in other words, it will give the lowest bit index that is not set in exactly one of b1 and b2). For example, for the sets {1, 2, 3, 4 } and {0, 3, 5, 6} the method will return 3 (the BitSet {1, 2, 3, 4} would be represented by the bit sequence 01111 and the BitSet { 0, 3, 5, 6} would be represented by the bit sequence 1001011).

You will receive full credit if your implementation does not use a loop. We have included at the end of the exam information about BitSet methods you can use for this problem.

Problem 4 Polymorphism (13 points)

Assume we have defined the following classes (we will plug something into ????? later):

```
public class A {
        public void f(A a) {
                System.out.println("A.f(A)");
        }
        public void f(B b) {
                System.out.println("A.f(B)");
        }
}
```

```
public class B extends A {
        public void f(A a) {
                System.out.println("B.f(A)");
        }
        public void f(B b) {
                System.out.println("B.f(B)");
        }
        public void g() {
                System.out.println("B.g()");
        }
}
```

```
class Test {
        public static void main(String[] args) {
                A a = new A();
                A ab = new B();
                B b = new B();
                ?????
        }
}
```

a) Give an example of method overloading in these classes.

b) Give an example of method overriding in these classes.

c) For each of the following things that might go where ????? occurs, write CE if it would generate a compile time error, write RE if it would generate a runtime error, and write down what the program would print if it would run and print something.

a.f(a);	
a.f(ab);	
a.f(b);	
ab.f(a);	
ab.f(ab);	
ab.f(b);	

b.f(a);	
b.f(ab);	
b.f(b);	
a.g();	
ab.g();	
b.g();	

Problem 5 Writing test cases (12 points)

Assume we wish to test the following two methods:

```
public class Util {
        /**
         * Return the minimum value (using the standard < ordering)
         * of the arguments
         */
        public static int min(int a, int b, int c) { ...  }

        /**
         * Return the index of the first location in a that contains
         * the same value as x. If x does not occur in x, return -1.
         */
        public static int indexOf(int a[], int x) { ...  }
}
```

Write a set of test cases to test each of these methods. Give each test case as an expression that should evaluate to true; you don't need to worry about creating jUnit tests, and don't use loops or other coding techniques to make a large number of calls. Just give a reasonable number of expressions to use to test these methods. Note that you can use, for example,

```
new int[] {5,3}
```

in an expression to create an int array of length 2 initialized to contain 5 and 3.

Honors Section Problem (10 pts)

Lots of problems have been proven to be undecidable: not only the halting problems, but the book described other undecidable problems such as the tiling problem and domino snakes on the half plane.

a) What general approach/technique might you use to show that a problem is undecidable?

b) Say you are working at a company on a project, and the feature set for your project includes solving a problem that are you able to show is undecidable. What are the practical implications of this?