Cornell net id _____ Name _____

Section day _____ Section time _____

CS 100J Prelim 2 *Have a good break!!!* 15 March 2007

This 90-minute exam has 6 questions (numbered 0..5) worth a total of 100 points. Spend a few minutes looking at all questions before beginning. Use the back of the pages if you need more space.

Question 0 (2 points). Fill in the information, legibly, at the top of each page (Hint: do it now.)

0 _____	out of 02
1 _____	out of 18
2 _____	out of 20
3 _____	out of 20
4 _____	out of 20
5 _____	out of 20
Total _____	out of 100

Question 1 (18 points)

(a) What is a loop invariant?

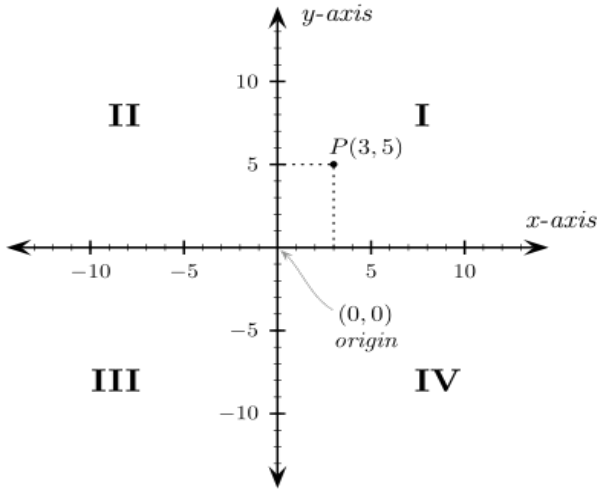
(b) Below is a loop. Fill in the invariant, then the initialization, then the loop body.

```
// n > 0.
// Assume that function f(int i) returns an int value.
// Store in m the maximum value of f(i) for i in the range 0..n-1.

int max=                                                    ;


// invariant:


for (int k= 1; k < n; k= k+1) {









}
// m = maximum value of f(i) for i in the range 0..n–1
```

Cornell net id _____ Name _____

Section day _____ Section time _____

Question 2 (20 points): Below is a diagram of the x-y plane. Each point in the plane is determined by its *x-coordinate* and *y-coordinate*. In the diagram, point P has x-coordinate 3 and y-coordinate 5.




```
/** instance is a pt in x-y plane */
public class Point {
   private int x; // x coordinate
   private int y; // y coordinate

   /** Constructor: Point with x
      coordinate a and y-coordinate b.  */
   public Point (int a, int b) {
         x= a;       y= b;
   }
   /** = x-coordinate */
   public int getX() { return x; }

   /** = y-coordinate */
   public int getY() { return y; }
}
```

An instance of class Point contains the x-coordinate and y-coordinate of a point. Note that an object of class Point is "immutable": there is no way to change its fields.

On the back of the previous page, write the body of the procedure that is specified below. If you write a loop, you need not write a loop invariant, although it may help you to do so.

/** Replace all points in v whose x- and y-coordinates are *both* negative by the corresponding points whose x- and y-coordinates are positive. E.g. replace a point (-3, -5) in v by point (3, 5). All other points in v remain unchanged. E.g. change

v = [(-3, 5), (-3, -5), (-2, -6), (3, 5)] to [(-3, 5), (3, 5), (2, 6), (3, 5)] */

public static void makePos(Vector< Point > v)

You can use the following methods:

	v.remove(**int** k)	Remove element v[k], changing v so that it contains v[0..k–1] followed by v[k+1..]
Assume v has type Vector<C> for some class C		
Return	**Method**	**Purpose**
C	v.get(**int** k)	= v[k].
int	v.size()	= the number of elements in v.
	v.set(**int** k, C ob)	Store ob in v[k].

Cornell net id _____ **Name** _____

Section day _____ **Section time** _____

Question 3 (20 points). Questions 3 and 4 deal with the three class definitions (of classes Animal, Elephant, and AsianElephant) that appear in the following two boxes:

```
public class Animal {
  private String name; // name of this animal
  private int weight;     // weight in pounds

  /** Constructor: animal with name n and
       weight w. */
  public Animal(String n, int w) {
     name= n;
     weight= w;
  }

  /** = name of this animal. */
  public String getName()
      { return name; }

  /** = weight of this animal, in pounds. */
  public int getWeight()
      { return weight; }

  /** = Type of this animal. */
  public String AnimalType()
      { return "Animal"; }

  /** = "Animal a is in v*/
  public static boolean isIn(Animal a,
                             Vector<Animal> v) {
     for (int k= 0; k < v.size(); k= k + 1) {
        Animal b= v.get(k);
        if (a == b) return true;
     }
     return false;
  }
}
```

```
public class Elephant extends Animal {

  private int ht; // elephant height in inches

  /** Constructor: instance with name n,
       weight w, and height h. */
  public Elephant(String n, int w, int h) {
     super(n, w);
     ht= h;
  }

  /** = height of this elephant. */
  public int getHeight()
      { return ht;}

  /** = Type of this animal. */
  public String AnimalType()
      { return "Elephant"; }
}


public class AsianElephant
                    extends Elephant {

  /** Constructor: instance with name n,
       weight w, and height w. */
  public AsianElephant(String n, int w, int h)
      { super(n, w, h); }

  /** = "Asian Elephant". */
  public String AnimalType()
      { return "Asian Elephant"; }
}
```

(a) What is the apparent type of a variable? What is the real type of a variable? What are the apparent and real types of variable v after execution of the following statement?

```
Animal v= new AsianElephant("Ganesha", 2000, 60);
```

Cornell net id _____ **Name** _____

Section day _____ **Section time** _____

(b) Draw a folder (object) of class AsianElephant. Do not include the partition for class Object.

(c) Each of the five cases below consists of a statement followed by an expression. Write the value of the expression after the statement is executed —if you think that execution would lead to an error, then explain the error. Remember, it often helps to draw objects and variables.

(1)
```
Elephant e= new AsianElephant("Elephas", 10000, 120);
"Elephant".equals(e.AnimalType())
```

(2)
```
Animal c= new Elephant("Maximus", 9000, 100);
c.AnimalType()
```

(3)
```
AsianElephant b= (AsianElephant)(new Elephant("Indicus", 8000, 90));
b.AnimalType()
```

(4)
```
Animal f= (Animal)(new Elephant("Loxodonta", 12000, 130));
f.getHeight()
```

Cornell net id _____ **Name** _____

Section day _____ **Section time** _____

Question 4. (20 points).

(a) Consider the statement shown below. Draw the frame for the call on static function isIn, where isIn is defined in class Animal on page 3. You do not have to assign argument values to parameters, and you do not have to execute the method body. We want to see only what the frame for the call looks like after it has been created.

```
Boolean b=  Animal.isIn(null, new Vector<Animal>());
```

(b) Write an instance method equals(Object obj) for class Elephant in Question 3. To help you out, here is the beginning of the class definition for Elephant, showing its one field and method equals, whose body you have to write.

```
public class Elephant extends Animal {
   private int ht; // elephant height in inches

   /** = "obj is an Elephant with the same values in its fields as this Elephant" */
   public boolean equals(Object obj) {




   }
}
```

Cornell net id _____ **Name** _____

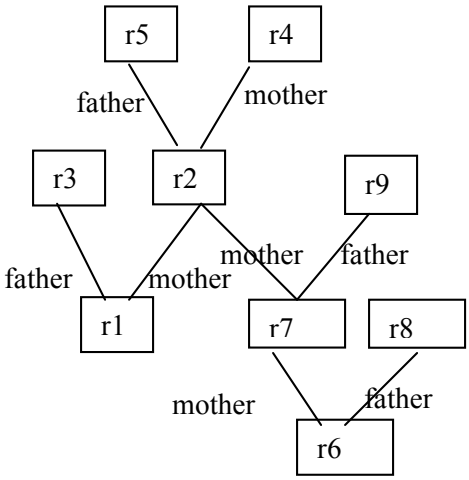
Section day _____ **Section time** _____

Question 5 (20 points). In Assignment A1 and A3, you implemented a class Rhino, part of which is shown below —we put only the fields and methods needed for this question.

The *family tree* of a rhino consists of that rhino, its known parents, their known parents, etc. The diagram to the right shows rhino r1's family tree, which consists of r1, r1's mother r2; r1's father r3, and r2's parents. Rhino r3's parents are unknown.

The diagram also shows r6's family tree.

Two rhinos are *related* if their family trees share a rhino. Thus, r6 and r1 are related, because their trees share rhinos r2, r4, and r5.



Write the body of static recursive function areRelated, which has been put into class `Rhino`. You may not use a loop; you must use recursion. Remember that two rhinos are related if there is some rhino that is in both of their trees. Also, if a Rhino variable is null, its family tree is empty.

In solving this problem, think of all possible base cases first and write Java code for them. Finally, in dealing with the recursive case, you will may need several recursive calls. Remember, the idea in the recursive case is to solve the original problem in terms of the same kind of problems but on a smaller scale.

```
public class Rhino {
    private Rhino father; // this Rhino's father (null if unknown)
    private Rhino mother; // this Rhino's mother (null if unknown)
    /** = Rhinos ra and rb are related */
    public static boolean areRelated (Rhino ra, Rhino rb) {









    }
}
```