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.)

**Question 1 (15 points).** Write the body of the following function.

Here are the ground rules:
1. Do not write any other method.
2. Use recursion; do not use a loop.
3. Do not declare a local String variable or use any String function.
4. You will have to use String operation *catenation* (+).

```java
/** = n, as a String, but with its digits reversed.
 * Precondition: n >= 0.
 *   e.g. if n = 135720, the value returned is "027531".
 *   e.g. if n = 12345, the value returned is "54321".
 *   e.g. if n = 7, the value returned is "7".
 *   e.g. if n = 0, the value returned is "0".*/

public static String rev(int n) {

}
```

16 October 2007
Question 2 (30 points): This question and the next deals with classes that maintain information about bees. On the bottom right of this page is a class Bee, with some parts of it not yet completed. Yes, we are going to have tags to distinguish bees, as strange as that may sound.

Answer the following questions.

A. Look at the second constructor, the one at the top of the righthand column. Notice that the new Bee object is supposed to be given a unique tag—in field tag. Your job is to implement the rest of the method body of the second constructor to accomplish this. You can declare a static variable in the class to help you accomplish this task, if you wish. If you do declare a variable, be sure to explain in a comment what it means.

B. Complete the body of the first constructor. The body should be a single statement. Writing more than one statement gets you at most half credit.

C. Suppose the following function, equals, is to be placed within class Bee. Complete the body of the function. Note that field tag should not be tested.

```java
/** = obj is an object of class Bee and has the same month and year of birth as this Bee. */
public boolean equals(Object obj) {

/** An instance represents a Bee */
public class Bee {

private int month; // month of birth
private int year; // year of birth
private int tag; // unique number >= 0.
    // No two Bees have the
    // same tag.

/** Constructor: A bee with birth month 1
 and birth year 0. The bee is given a
 unique tag. */
public Bee() {

} //continued in next column
```
/** Constructor: A bee with birth month m and birth year y. The bee is given a unique tag. Precondition: 1 <= m <= 12. */
public Bee(int m, int y) {
    month= m;
    year= y;
}

/** = this Bee's year of birth */
public int getYOB() { return year; }

/** = this Bee's month of birth */
public int getMOB() { return month; }

/** = this Bee's tag (-1 if none) */
public int getTag() { return tag; }
Question 3 (18 points). Did you know that a male bee has a mother but not a father? But a female bee has both a father and a mother. A queen bee is represented by an object of class Bee (previous question), and we don’t know her mother and father. A male bee is represented by an object of class MaleBee, shown below. A female bee that is not a queen is represented by an object of class FemaleBee, shown below. All parents of all bees are known, except for queens.

To the right is part of a family tree. Parents of queens (q1, q2, q3) are not known. Males (m1, m2) have only a mother. Non-queen females (f1, f2) have both a mom and a dad.

Write the body of the following function. Use recursion. Do not loops. An ancestor is a mother, father, grandmother, grandfather, etc.

Besides the necessary recursion, think also about issues like these: (1) How do you tell the base case (i.e. that b is a queen)? (2) If b is not a queen, how do you cast it to the proper class so that its methods can be used?

Use the back of another page if you want. You can separate the pages.

```java
/** = number of female ancestors of bee b.
 * Precondition: b is not null. */
public static int femAnc(Bee b) {
    // Function body
}

/** A male bee */
public class MaleBee extends Bee {
    private Bee mother; // mother of this bee
    /** Constructor: A male bee with birth date
     * month and year and mother mom.
     * Precondition: mom is not null. */
    public MaleBee(int month, int year, Bee mom) {
        super(month, year);
        mother = mom;
    }
    /** = the mother of this bee */
    public Bee getMother() { return mother; }
}

/** A female bee (that is not a queen) */
public class FemaleBee extends Bee {
    private Bee mother; // mother of this bee
    private MaleBee father; // father of this bee
    /** Constructor: A female bee with birth date
     * month/year, mother mom, and father dad.
     * Precondition: mom and dad are not null. */
    public FemaleBee(int month, int year,
                     Bee mom, MaleBee dad) {
        super(month, year);
        mother = mom;
        father = dad;
    }
    /** = the mother of this bee */
    public Bee getMother() { return mother; }
    /** = the father of this bee */
    public Bee getFather() { return father; }
}
```
Question 4. (15 points).

(a) Write the four steps in evaluating a function call.

(b) To the right is part of class Bee of a previous question, with just enough of it for you to answer this question and an additional function isOlder.

Suppose variable b1 and b2 contain the names of Bee objects, as shown here:

\[
\begin{array}{c}
\text{b1} & \text{a5} & \text{b2} & \text{a8}
\end{array}
\]

Draw the frame for the following call, including its scope box.

\[
b1.isOlder(b2)
\]
Question 5 (20 points).  

(a) Consider this function.

```java
public void f(int p) {
p = p + 1;
if (p < 0) {
    int k = p;
    p = p + k;
}
}
```

When are parameter `p` and local variable `k` created?

(b) What is an argument?

(c) Consider the following statement, where class `Bee` is given on page 2 and class `MaleBee` on page 3:

```java
Bee b = new MaleBee(5, 2007, bmom);
```

What are the apparent and real classes of `b` after execution of this statement?

(d) Suppose `b` using the assignment in part (c). Indicate which of the following three expressions are syntactically legal or illegal. For an illegal one, explain why it is illegal; then, if is possible to change it so that it is legal (and returns the obviously desired value), do so; if it is not possible, explain why.

1. `b.getTag()`
2. `b.getMother()`
3. `b.getFather()`