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

**Question 0 (1 point).** Fill in the information, legibly, at the top of each page. (Hint: do it now.)

**Question 1 (20 points).**

(a) Evaluate the call `Numbers.f(1)` (see the class at the bottom of the page). Note that we have labeled each statement to help you identify them. As you evaluate the function call, draw frames for all calls that are evaluated. Stop executing just after executing one of the statements labeled `p2` and `p3`, whichever comes first. Do not continue executing. That way, we will be able to see that you drew frames and executed correctly.

(b) When is parameter `i` created? When is local variable `k` created? When is local variable `s` created?

```java
public class Numbers {

    public static int f(int i) {
        int k;
        f1: k = p(i + 1);
        f2: return k;
    }

    public static int p(int n) {
        int s;
        p1: if (n % 2 != 1) {
            p2: s = n * n;
        }
        else {
            p3: s = n;
        }
        p4: return s;
    }
}
```
Question 2 (32 points). Write the two functions $fa$ and $pow$. They are defined below in class Rhino—we put only the fields and methods needed for this question. Don’t use loops; you must use recursion. Don’t write any other methods.

The following will help you understand function $fa$. The diagram to the right shows Rhino $r1$’s family tree; $r1$ has a mother and a father, and $r1$’s father has a mother, so $fa(r1)$ is 2.

In the second function, do not be concerned with efficiency; the number of recursive calls may be proportional to $p$.

In solving both problems, think of the base case first and write Java code for it. In dealing with the recursive case(s), remember that the idea is to solve the original problem in terms of the same problem but on a smaller scale.

```java
public class Rhino {
    private Rhino father; // this Rhino’s father (null if unknown)
    private Rhino mother; // this Rhino’s mother (null if unknown)

    /** = number of female ancestors of r (i.e. the number of mothers, grandmothers, 
     * great grandmothers, etc. Note that r may be null. If it is, use 0 for the answer. */
    public static int fa(Rhino r) {
        // Your code here
    }

    /** = n**p (that is, n to the power p).
     * Note: p can be negative, 0, or positive. n**0 = 1. For p negative, n**p = 1/(n**-p).
     * Examples: pow (2, 1) = 2. pow(2, -1) = 0.5.
     * pow (2, 2) = 4. pow(2, -2) = 0.25. */
    public static double pow(double n, int p) {
        // Your code here
    }
}
```
Question 3 (18 points) We ask you to write a loop with initialization. You must use the command, post-condition, and loop invariant that is given. Thus, the initialization and loop must be consistent with the invariant. You are not writing a complete method, so do not write a return statement.

Assume that Vector v is declared as follows and that v contains at least one element.

```java
Vector<String> v;
```

Complete the initialization, the loop initialization, loop condition, loop increment, and repetend for the following —read the invariant carefully!

```java
// Assume v contains at least one element and x is declared.
// Store in x the length of the longest String in v.

// invariant: 1 <= k &&
//           x = length of the longest string in v[0..k-1]

for (int k = ; ; ) {

}

// x = length of longest string in v
```

Useful operations on a Vector v:
- v.size() number of elements in v.
- v.get(k) element number k of v, or v[k]
Question 4. (20 points). In answering the parts of this question, it should help to draw the objects that are created and the frames for the function calls that are being evaluated. You don’t have to do this, but it should help.

Consider the two classes given at the bottom of this page. Suppose the following assignments have been executed (we also give the declarations of the variables being assigned):

```java
One a;  a= new One(2);
One b;  b= new One(3);
One c;  c= new Two(4);
```

(a) Evaluate the following expressions and write their values to the right of the expressions.

1. One.sum(a, b)
2. One.sum(b, c)
3. One.sum(a, c)
4. a instanceof One
5. a instanceof Two
6. c instanceof One
7. c instanceof Two
8. c.alternateMult()

(b) In the expression One.sum(a, c):

1. What are the apparent and real classes of a?
2. What are the apparent and real classes of c?

```java
public class One{
    public int baseValue;

    public One (int n) {
        baseValue= n;
    }

    public int mult() {
        return 1*baseValue;
    }

    public static int sum(One a, One b) {
        return a.mult() + b.mult();
    }
}

public class Two extends One {
    public Two(int n) {
        super(n);
    }

    public int mult() {
        return 2*baseValue;
    }

    public int alternateMult() {
        return super.mult();
    }
}
```
Question 5 (9 points).

(a) Define parameter and argument.

(b) Suppose class `C` has a constructor with this specification:

```java
/** Constructor: An instance with temperature \( t \) on day \( d \) */
public C(double \( t \), Day \( d \))
```

Complete the body of the following constructor, using exactly one statement and no more.

```java
/** Constructor: An instance with temperature 0 on day \( d \) */
public C(Day \( d \)) {
}
```

(c) Suppose variable \( v \) is declared as

```java
Animal \( v; \)
```

Write an expression to cast it to class `Cat`. Also, explain under what condition the cast will not produce an error.