CS100J Prelim II, 21 Oct. 2003 CORNELL NETID _____

NAME (PRINT LEGIBLY!)_____
(last, first)

This 90-minute exam has 8 questions (numbered 0..7) worth a total of 100 points. We suggest that you spend a few minutes looking at all questions before beginning so that you can see what is expected. Budget your time wisely. Use the back of the pages, if you need more space.

Question 0 (2 points). Write your netid and your name, legibly, at the top of each page.

Question 1 (18 points). (a) Write a declaration for a local String variable `line`. It should not give an initial value to `line`.

(b) Write a sequence of Java statements to print the number of characters at the beginning of `line` that are the same as the first character in `line` . Assume that `line` contains at least 1 character.

Don't write a complete method; just write the statements. Declare any variables that you use (except variable `line`). You will need to write a loop. You don't have to write an invariant if you don't want to. Remember that `line.charAt(k)` yields the character at index `k`. Examples:

If `line` is `"aaabcd"`, the output is `3`.

If `line` is `"abcd"`, the output is `1`.

If `line` is `"a"`, the output is `1`.

If `line` is `"aabaaa"`, the output is `2`. (Later occurrences of the same character do not count.)

If `line` is `"aaAaaa"`, the output is `2`. (Upper-case and lower-case are different.)

Cornell net id _____ **Name** _____

Question 2. (12 points). Write down the sequence of steps that are performed in executing the procedure call `C.meth(exp);` on a static procedure `meth` in class `C`. Be sure to say what goes in the scope box.

Question 3. (16 points) Below is a precondition, loop invariant, postcondition, and outline of a while-loop. Fill in the initialization, loop condition, and repetend. Your grade depends entirely on how well you deal with the four loopy questions. The algorithm being written is looking for the largest power of 2 that is not larger than n. Using the notation d**c for d multiplied by itself c times (so 2**3 = 8 and 2**0 = 1), we give some examples:

For n = 9, b ends up at 8, since 8 = 2**3 <= 9 and 2**4 > 9.

For n = 15, b ends up at 8, since 8 = 2**3 <= 15 and 2**4 > 15.

For n = 16, b ends up at 16, since 16 = 2**4 <= 16 and 2**5 > 16.

// precondition: n > 0

// invariant: b is a power of 2 and b <= n

while ( ) {

}

// postcondition: b is a power of 2 and b <= n and n < 2*b

Cornell net id _____ **Name** _____

4. (16 points) Below is *part* of a class whose instances are fractions, like 3/4, 12/4, 0/8, and –5/4. We view fractions 3/4, 6/8, and 12/16 as being equal, as usual.

Note the "class invariant", which describes conditions that fields `numerator` and `denominator` must adhere to.

Write the bodies of the constructor (note carefully its specification) and function `equals`. In writing them, you do not have to check that preconditions mentioned in the specs are met. You do not have to write the body of procedure `reduce`, but you may want to use procedure `reduce`.

The following property of fractions may be useful to you: b / c = –b / –c. There are other such properties that might be useful.

```
// An instance is a fraction numerator/denominator   (e.g. –5/3 and 5/15)
public class Fraction {
    /** Class invariant: the fraction is numerator/denominator.
        The numerator may be any int, but the denominator is always > 0. */
    private int numerator;
    private int denominator;

    /** Constructor: the fraction a / b. Precondition: b != 0.
    public Fraction (int a, int b) {









    }

    /** Reduce this fraction to its lowest terms. For example, 12/16 would be reduced to 3/4. */
    public void reduce () { // Details omitted. You do NOT have to complete this method. }

    /** = "this fraction equals fraction otherFrac". Note, for example, that the fraction 2/4 equals
            the fraction 8/16, so be careful how you write this function. */
    public boolean equals (Fraction otherFrac) {









    }
}
```

Cornell net id _____ **Name** _____

Question 5. Subclasses (14 points).

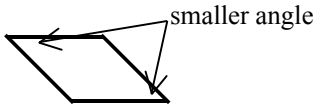
To the right is a class Rhombus, each instance of which represents a rhombus, using its side length and smaller angle. (A rhombus is a 4-sided figure with all four sides equal. An example appears below the class.) Write a subclass Square of Rhombus whose instances represent squares. Here are its properties:

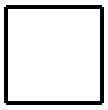
(0) Subclass Square should NOT declare any fields.
(1) Its constructor should have 1 parameter, the length of each of it sides.
(2) It inherits methods of Rhombus.
(3) It should define a new instance method "area" that returns the area of the instance in which it appears.

```
public class Rhombus {
  private int  side;   // length of each side
  private int  angle; // smaller angle
  // Constructor: rhombus with side length s
  //  and smaller angle a
  public Rhombus(int s, int a)
     {side= s;  angle=  a;}

  // =  length of a side of this rhombus
  public int getSide() {return side;}

  // = smaller angle of this rhombus
  public int getAngle() {return angle;}
}
```





rhombus
all 4 sides equal
opposite angles equal

square
4 sides equal
angles are 90 degrees

Question 6. Subclasses —refer to Question 5. (12 points) Below, show the contents of the variables after all four statements that are shown below have been executed. (You do **not** have to show the frames for the method calls but you **do** have to draw folders). If a statement is illegal, say so and leave the variable being assigned empty. We show the 4 variables into which you should place values.

```
Square x= new Square(5);
int a= x.area();
Rhombus y= new Square(6);
int b= y.area();
```

x ☐ a ☐

y ☐ b ☐

Cornell net id _____ **Name** _____

Question 7 (10 points).

0	_____ out of 02
1	_____ out of 18
2	_____ out of 12
3	_____ out of 16
4	_____ out of 16
5	_____ out of 14
6	_____ out of 12
7	_____ out of 10
Total	_____ out of 100

(a) Define "repetend", as used in discussing while-loops.

(b) Write down the four loopy questions.