1. Fill in the body for the following method. Use constants (i.e. final local variables) where appropriate.

```java
/** Converts a weight from metric to avoirdupois (i.e., US ounces and pounds) and prints a string of the form "500 grams is 1 pound and 1.636981 ounces" to System.out. Note that there are 28.3495231 grams per ounce and 16 ounces in a pound.

@param grams the weight to convert, in grams
*/

public static void gramsToAvoirdupois(int grams) {
    // implementation
}
```

2. Examine the following method:

```java
public void foo(float f) {
    int x = (int)f;
    int y = Math.round(f);
    // SAME
}
```

Will x and y always have the same value at the point marked SAME? Why or why not?

3. What is a reasonable precondition (i.e. a requirement imposed upon the arguments) for the following method?

```java
public static double reciprocal(int x) {
    return 1d/x;
}
```

4. The distance between two points on a plane (x1, y1) and (x2, y2) can be calculated as follows:

```
distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}
```

Write a method that takes four double values corresponding to two pairs of points and calculates the distance between them. Use appropriate static methods from the Math class.
5. Consider the following program fragment:

```java
int x = 2147483647;
x++;
```

What is the value of `x` at the end of this fragment? Explain your answer. Does the illustration below have anything to do with this problem?

6. What output does the following program fragment produce? Explain your answer.

```java
System.out.println('A' + 'B' + 'C');
System.out.println("A" + "B" + "C");
System.out.println("A" + 'B' + 'C');
System.out.println("A" + ('B' + 'C'));
```

7. Many applications require statistical analysis of very large streams of data. In many cases, we would like to know information about the data in the aggregate (e.g. min/max values, arithmetic mean), but it is not feasible to store each individual data sample. For this problem, you will implement a `SampleTracker` class with the following public interface:

```java
public class SampleTracker {
    public void addSample(int x);
    public int getSampleCount();
    public int getMin();
    public int getMax();
    public double getMean();
}
```

A skeleton class file with JavaDoc comments and more hints is available on the web from:

http://www.cs.wisc.edu/~willb/cs302

Hint: use the `Math.min(int x, int y)` and `Math.max(int x, int y)` methods. Do not use conditional statements, and do not keep track of each individual sample.