1) We know that arrays are nice to use, as long as we aren't walking off their ends. What line would you write to print the Q in the array below? What about the G? What about all the letters, in the order they're listed below?

```
char [] some_letters = {'G', 'W', 'S', 'Q'};
```

2) What does a for loop do? It's a specialized and more compact version of a while loop. In either kind of loop, we need to:

1. Initialize some variable to control how many times the loop executes
2 Identify a termination condition that is true for some time (while the loop executes) and becomes false at some time (so the loop eventually terminates.
3. Provide some way to update the control variable.

Here's a while loop:

```
k = 100;
while (k > 0)
{
        System.out.println(k);
        k = k – 2;
}
```

Rewrite this as a for loop.

3) For loops are especially useful with arrays, because Java knows how long they are. That's why it's nice to use one in QuantMisplacedTiles, to loop over the tile array and count up a running sum of the Manhattan distance for all the tiles. You should keep this code in mind for the exam, which has a similar problem. Here's another example. Write code that counts up how many odd numbers are in the array.

```
int odd_count;
int [] some_numbers = {3, 17, -8, 2, 4};
```

4) You may have noticed that semicolons don't belong on lines after if, while, or for statements. The compiler doesn't mind extra semicolons, but it interprets them as empty statements. What will this code do? Why?

```
j = 0;
if (j > 10);
{
        System.out.println("j is larger than 10");
}
```

5) When we talked about classes, we said that some class variables are public and others are private. What's the difference? As a practical example, what might be wrong with the code below? Also, what would happen if the constructor were a private method?

```
public class Point
{
        public int x;
        private int y;
        public Point(int start_x, int start_y)
        {
                x = start_x;
                y  = start_y;
        }
}

public static void main (String [] args)
{
        Point mypoint = new Point(3,2);
        mypoint.x = 10;
        mypoint.y = 9;
}
```

6) What's the relationship between a class and an object? A class, like Point above, defines a point in 2-dimensional space. When we declare a variable like mypoint, above, and define the x and y points, then we have a particular example, or object, of type Point. In the 8-puzzle code, is currentboard an object or a class? How do you know?

7) The other distinction you should understand is between static and non-static methods. Static methods seem weird, but they make sense to use when we write classes that don't translate naturally into objects. In Java, mathematical functions, like square root or absolute value, are all defined as methods of the Math class. But there is no corresponding object of type Math—who knows what a variable of type Math would look like? This raises a logical question: if no object exists for a particular class, how do we call the methods of that class? The answer is that we define these methods as static, and then call them through the class itself. For example, here's some code that makes two static method calls using the Math class methods:

```
float q = -10.5;
System.out.println(Math.sqrt(Math.abs(q)));
```

Suppose that we write a method in the Point class we defined a minute ago to move Points around by a distance dx and dy. It only makes sense to move particular points, with particular coordinates (because we need to know where a particular point is to move it). So MovePoint is not a static method, and it gets called via a particular point object. PrintType, however, is a static method, and it gets called via the class name, not the object name. The code below works and has an example.

```
public class Point
{
```

```
        public int x;
        private int y;
        public Point(int start_x, int start_y)
        {
                x = start_x;
                y  = start_y;
        }
        public MovePoint(int dx, int dy)
        {
                x += dx;
                y += dy;
        }
        public static PrintType()
        {
                System.out.println("I am a Point!");
        }
}

public static void main (String [] args)
{
        Point mypoint = new Point(3,2);
        mypoint.MovePoint(10, 9);
        Point.PrintType();
}
```

8) In the 8-puzzle code, the PuzzleBoard methods aren't static, because it only makes sense to move tiles and check configurations for particular PuzzleBoard objects. But the CleverSearch methods are all static, and that's why we call them via the class name:

```
CleverSearch.SmartSearch(open, closed);
```

When we sort the list of open boards for the smart 8-puzzle search, we sort by their depth and by their score. This makes the search optimal. What does optimal mean in this context, and how does the way we sort the boards make the search optimal?

When we score 8-puzzle boards, why don't we care how far the blank tile is out of place? Will failing to count the misplaced tile interfere with our ability to detect perfect boards?