CSE 142 Section Handout #5
Style Sheet

This program prompts the user for a value and then generates random numbers between 0 and 99 until it finds a number divisible by the given number, as in:

```
number to use? 10
Let's find a number divisible by 10
83, 88, 32, 8, 33, 21, 95, 76, 89, 78, 2, 99, 1, 20
found one after 14 tries
```

Consider the following implementation of the program:

```
import java.util.*;

public class Sect5 {
   public static void main(String[] args) {
      Scanner console = new Scanner(System.in);
      System.out.print("number to use? ");
      int a = console.nextInt();
      gN(console, a);
   }

   // generates random numbers, printing numbers separated by commas
   public static void gN(Scanner console, int a) {
      System.out.println("Let's find a number divisible by " + a);
      int number = 1;
      int count = 0;
      while (number % a != 0) {
         Random r = new Random();
         number = r.nextInt(100);
         System.out.print(number);
         if (number % a != 0) {
            count++;
            System.out.print(", ");
         } else if (number % a == 0) {
            count++;
            System.out.println();
            System.out.println("found one after " + count + " tries");
         }
      }
   }
}
```

While this program would receive full external correctness by producing the desired output, it would not receive full internal correctness. List all style issues you can find.

CSE 142 Section Handout #5
Style Sheet Solutions

```
import java.util.*;

public class Sect5 {
   public static void main(String[] args) {
      Scanner console = new Scanner(System.in);
      System.out.print("number to use? ");
      int target = console.nextInt();
      generateNumbers(target);
   }

   // generates random numbers between 0 and 99 until it finds one divisible by
   // target, reporting a count of how many numbers were generated.
   public static void generateNumbers(int target) {
      System.out.println("Let's find a number divisible by " + target);
      Random r = new Random();
      int number = r.nextInt(100);
      int count = 1;
      while (number % target != 0) {
         System.out.print(number + ", ");
         number = r.nextInt(100);
         count++;
      }
      System.out.println(number);
      System.out.println("found one after " + count + " tries");
   }
}
```

- **Unnecessary parameter** – The Scanner parameter in the `generateNumbers` method is never used, and so is unnecessary. You should only pass parameters that are necessary to the functioning of the method.
  - Unnecessary parameters and misuse of parameters may lose style points in Homework 5.
- **Method comment** – Method comments need to address the purpose of the method, if applicable, the parameters that the method accepts, and if applicable, what the method returns.
  - Failing to comment on these things may result in a point loss in Homework 5.
- **Unnecessary object construction** – "It is good practice to minimize the number of objects that your program creates." –CSE142 Style & Commenting Guide. The `Random` object construction being placed inside of the `while` loop means that a brand new `Random` object was created in every iteration of the loop. By placing the initialization before the loop, only one `Random` object is created each time the method is called.
  - Creating objects unnecessarily may be a deduction in Homework 5.
- **Loop zen** – "Remember that loops should only be used for *repeated* tasks; if you only need to do a task once, then there's no need to introduce a loop in the first place." –CSE142 Style & Commenting Guide. The two `println`s at the end of the method are only executed once, and so don't need to be included in the `while` loop.
  - Poor loop zen may result in loss of style points in Homework 5.
- **If/else structure** – "When using conditional execution in your program, you should choose which combination would be best based on the minimum/maximum number of branches that could execute." –CSE142 Style & Commenting Guide. Exactly one branch is going to execute, so this structure should end in an '`else`' branch.
  - Incorrect conditional structure may result in point deductions in Homework 5.
- **If/else factoring** – "If you have common code being repeated between branches, then this is saying that you want to execute that code regardless of what you are testing for." –CSE142 Style & Commenting Guide. The `count++;` statement occurs in both branches of the conditional structure, and so should be factored out. Though, it actually turns out that this structure is unnecessary with our fencepost solution (see below).
  - Appropriate `if`/`else` factoring is worth points in Homework 5.
- **Fencepost solution** – Notice in the output of the method, there is a fenceposting issue (`n` numbers are printed, but only `(n - 1)` commas). By printing the last number outside of the `while` loop, the amount of code needed in the method is reduced and more clean.
- **Non-descriptive names** – Descriptive variable and method names should be utilized at all times to improve readability of your program.
  - Non-descriptive naming in your submission may result in point deduction from your grade in Homework 5.