3. Debugging A lecture

Consider a static method called **`findNumber`** that generates random numbers looking for a particular target a particular number of times. The method takes two parameters:

- `int target` - the number to search for
- `int count` - the number of times to find the target

The method generates and prints random integers between 1 and 10 (both inclusive) until the number `target` has been generated `count` times. It then returns how many numbers were generated.

For example, suppose the following call was made:

```
int tries = findNumber(5, 1);
```

This call might produce output like the following:

```
Searching for 1 5s
Generating numbers: 1 8 10 5
```

In the example above, at the end of this call, the value 4 would be returned from the method and stored in the variable `tries`, because 4 numbers were generated before one 5 was seen. (Note that, due to randomness, this call would always not produce the exact same output or return value.)

Consider the following proposed buggy implementation of `findNumber()`:

```
1  public static int findNumber(int target, int count) {
2      Random rand = new Random();
3      int found = 0;
4      int total = 0;
5
6      System.out.println("Searching for " + count + " " + target + "s");
7      System.out.print("Generating numbers: ");
8      while (found < count) {
9          int num = rand.nextInt(10) + 1;
10         System.out.print(num + " ");
11         if (num == count) {
12             found++;
13         }
14         total++;
15     }
16     System.out.println();
17
18     return total;
19 }
```

This implementation contains a single bug that is causing it to not work as intended.

(continued on next page...)

As an example, if the following code is executed:

```
int tries = findNumber(5, 1);
```

The buggy implementation might produce the following output:

```
Searching for 1 5s
Generating numbers: 2 9 10 8 1
```

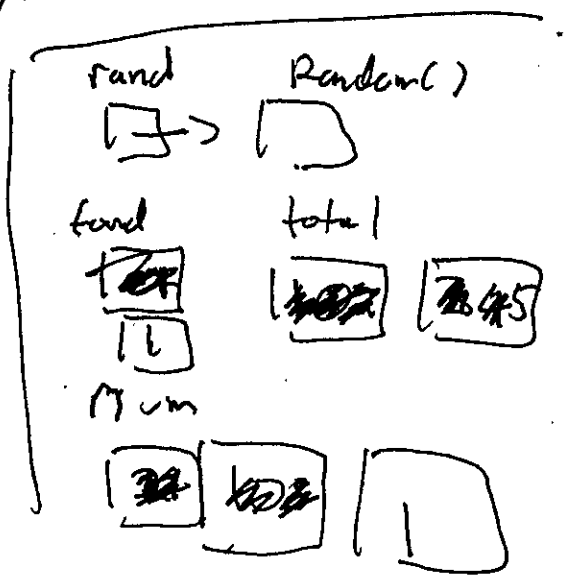
After this call to the buggy implementation, the variable `tries` would contain the value 5.

Part A: Identify the single line of code that contains the bug. Write your answer in the box to the right as a single number.

11

Part B: Annotate (write on) the code below to indicate how you would fix the bug. You may add (using arrows to indicate where to insert), remove (by crossing out), or modify (with a combination) any code you choose. However, the fix should not require a lot of work.

```
1  public static int findNumber(int target, int count) {
2      Random rand = new Random();
3      int found = 0;
4      int total = 0;
5
6      System.out.println("Searching for " + count + " " + target + "s");
7      System.out.print("Generating numbers: ");
8      while (found < count) {
9          int num = rand.nextInt(10) + 1;
10         System.out.print(num + " ");
11         if (num == target) {
12             found++;
13         }
14         total++;
15     }
16     System.out.println();
17
18     return total;
19 }
```



3. Debugging 13 lecture

Consider a static method called **`findNumber`** that generates random numbers looking for a particular target a particular number of times. The method takes two parameters:

- `int target` - the number to search for
- `int count` - the number of times to find the target

The method generates and prints random integers between 1 and 10 (both inclusive) until the number `target` has been generated `count` times. It then returns how many numbers were generated.

For example, suppose the following call was made:

```
int tries = findNumber(5, 1);
```

This call might produce output like the following:

```
Searching for 1 5s
Generating numbers: 1 8 10 5
```

In the example above, at the end of this call, the value 4 would be returned from the method and stored in the variable `tries`, because 4 numbers were generated before one 5 was seen. (Note that, due to randomness, this call would always not produce the exact same output or return value.)

Consider the following proposed buggy implementation of `findNumber()`:

```
1  public static int findNumber(int target, int count) {
2      Random rand = new Random();
3      int found = 0;
4      int total = 0;
5
6      System.out.println("Searching for " + count + " " + target + "s");
7      System.out.print("Generating numbers: ");
8      while (found < count) {
9          int num = rand.nextInt(10) + 1;
10         System.out.print(num + " ");
11         if (num == count) {
12             found++;
13         }
14         total++;
15     }
16     System.out.println();
17
18     return total;
19 }
```

This implementation contains a single bug that is causing it to not work as intended.

(continued on next page...)

As an example, if the following code is executed:

```
int tries = findNumber(5, 1);
```

The buggy implementation might produce the following output:

```
Searching for 1 5s
Generating numbers: 2 9 10 8 1
```

After this call to the buggy implementation, the variable `tries` would contain the value 5.

Part A: Identify the single line of code that contains the bug. Write your answer in the box to the right as a single number.

11

Part B: Annotate (write on) the code below to indicate how you would fix the bug. You may add (using arrows to indicate where to insert), remove (by crossing out), or modify (with a combination) any code you choose. However, the fix should not require a lot of work. 5, 1

```
1  public static int findNumber(int target, int count) {
2      Random rand = new Random();
3      int found = 0;
4      int total = 0;
5
6      System.out.println("Searching for " + count + " " + target + "s");
7      System.out.print("Generating numbers: ");
8      while (found < count) {
9          int num = rand.nextInt(10) + 1;
10         System.out.print(num + " ");
11         if (num == count) {
12             found++;
13         }
14         total++;
15     }
16     System.out.println();
17
18     return total;
19 }
```

print: 2.

target 5

count 1

found 0 1

total 5

rand