New York University
Department of Computer Science

CS101-004– Practice Problems
Intro to Computer Science

Instructor: Tobias Blickhan

-

Name: _____
Net ID: _____

---

1. You are given the following abstract class:

```
1 public abstract class RandomNumberGenerator {
2     protected int state;
3     public RandomNumberGenerator(int seed) {
4         state = seed;
5     }
6     public abstract int nextInt();
7     public abstract boolean nextBoolean();
8     public abstract double nextDouble(); // returns a number in [0, 1)
9 }
```

   (a) One way to generate pseudo-random numbers is to perform multiplication and modulo operations. Implement a subclass `MLCG` of `RandomNumberGenerator` that uses the following formula to generate the next random number integer:

$$x_{n+1} = (a \cdot x_n) \mod m, \tag{1}$$

   where $a = 2^{31} - 1$ and $m = 16807$. x_0 is the seed value passed to the constructor. Use `Math.pow(2, 31)` to calculate a but remember this returns a `double`.

   - All `MLCG` objects should share the same a and m values and their value should not change.
   - The state of the random number generator should change such that you can repeatedly call `nextInt()` to get a new random number. The state at iteration n is x_n.
   - You need to write three methods and a constructor, the latter takes a seed (int) and relies on the constructor of the superclass.

   Hint: What is the largest value that x_{n+1} can take?

   (b) The seed passed to the constructor should be positive and strictly smaller than m. Re-write the constructor so that it throws an `IllegalArgumentException` if it is not.

   (c) Your random number generator works fine, but after a while, it starts returning negative numbers, even though $(a \cdot x_n) \mod m$ should always be positive. What is going on?

   (d) Assume now that this issue is fixed. Next, write a method that takes any (!) RandomNumberGenerator and an integer n. (This method would be in a separate test class.) The method then generates n random numbers between 0 and 1. While doing so, it counts the number of values between 0 and 0.1, 0.1 and 0.2, and so on. There are ten such "bins". Return an array of length 10 with the counts.