WIA2005 Algorithm Design &Analysis
Semester 2, 2016/17
Lab 4

PART 1 – Implementation

1. Perhaps the best way of generating "random" numbers is by the linear congruential method:
   - r = (a * r + b) % m;
     where a and b are large prime numbers, and m is 2^{32} or 2^{64}
   - The initial value of r is called the seed
   - If you start over with the same seed, you get the same sequence of "random" numbers

   One advantage of the linear congruential method is that it will (eventually) cycle through all possible numbers. Almost any "improvement" on this method turns out to be worse. "Home-grown" methods typically have much shorter cycles.

   Write a code to produce a random number based on the above.

2. The following methods are needed to find a random number.

- import java.util.Random;
- new Random(long seed) // constructor
- new Random() // constructor, uses System.timeInMillis() as seed
- void setSeed(long seed)
- nextBoolean()
- nextFloat(), nextDouble() // 0.0 $\leq$ return value < 1.0
- nextInt(), nextLong() // all 2^{32} or 2^{64} possibilities
- nextInt(int n) // 0 $\leq$ return value < n
- nextGaussian()
  - Returns a double, normally distributed with a mean of 0.0 and a standard deviation of 1.0

This is an example code:

```
import java.util.*;

public class RandomDemo {

   public static void main( String args[] ){

   // create random object

   Random randomno = new Random();

    // check next int value

   System.out.println("Next int value: " + randomno.nextInt(10000));

   }

}
```

Write similar code using other methods next.Boolean, next.Folat() and ect.

3. The following code is a bad way of suffling an array. Rewrite the code so that all permutation (of the numbers) are equally likely.

```
static void shuffle(int[] array) {
    for (int i = 0; i < array.length; i++) {
        int j = random.nextInt(array.length);
        swap(array, i, j);
    }
  }
```