Experiment No. 1

Title: Random Number Generator

Batch: B2 **Roll No.: 16010421059** **Experiment No.:01**

Aim: To study and implement a PseudoRandom Number Generator (PRNG) using Linear Congruential Method

---

Resources needed: Turbo C / Java / python

Theory

Problem Definition:
Write a Program for generating random numbers using Linear Congruential method such that
i) Period of the numbers generated is >=100
ii) Density of the numbers generated is maximum (average gap between random numbers is < 0.1).

Concepts:

Random Numbers: Random numbers are a necessary basic ingredient in simulation of almost all discrete systems. Most computer languages have a subroutine, object or function that will generate a random number. A simulation language generates random numbers that are used to generate event times and other random variables.

Properties of random Numbers:
A sequence of random number R1, R2 … must have two important statistical properties, uniformity and independence.
Uniformity :
If the interval (0, 1) is divided into „n" classes or subintervals of equal length , the expected number of observations in each interval is N/n, where N is total number of observations.
Independence:
The probability of observing a value in a particular interval is independent of the previous drawn value.

Problems faced in generating random numbers:
1. The generated number may not be uniformly distributed.
2. The number may be discrete valued instead of continuous values.
3. The mean of the numbers may be too high or low
4. The variance of the number may be too high or low.
5. The numbers may not be independent
   e.g. a. Autocorrelation between numbers
   b. Numbers successively higher or lower than adjacent numbers.

Criteria for random no. generator:
1. The routine should be fast.
2. The routine should be portable.
3. The routine should have a sufficient long cycle. The cycle length or period represents the length of random number sequence before previous numbers begin to repeat themselves in an earlier order. A special case of cycling is degenerating. A routine degenerates when some number appears repeatedly which is unacceptable.
4. The random number should be replicable.
5. Most important, the generated random numbers should closely approximate to the ideal statistical properties of uniformity and independence.

Procedure / Approach /Algorithm / Activity Diagram:

Linear Congruential Method:

The Linear Congruential method produces a sequence of integers X1, X2,… between 0 and m-1 according to the following recursive relationship.

$X_{i+1} = (a X_i + c) \bmod m$, i = 0, 1, 2…

The initial value X_0 is called the seed, a is constant multiplier, c is the increment and m is the modulus. Maximal period can be achieved by a, c, m, X_0 satisfying one of the following conditions

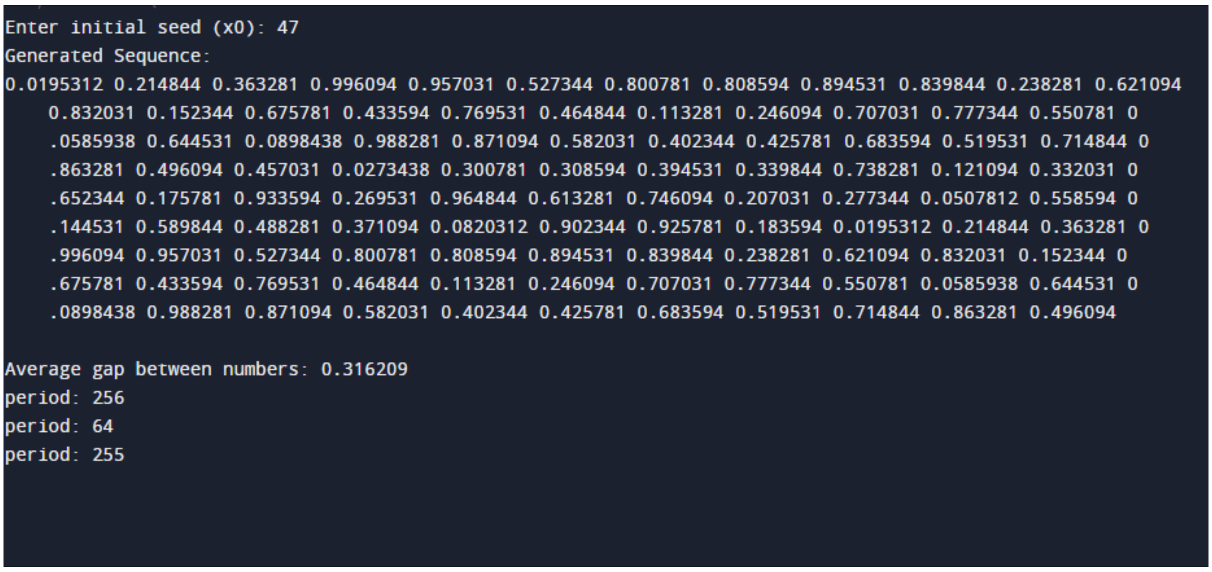
1. For m, a power of 2 ($m = 2^b$) and $c \neq 0$ period $p = 2^b$ is achieved provided c is relatively prime to m and a = 1+4k , k = 0,1,2,…
2. For $m = 2^b$ and c = 0 , period $p = 2^{b-2}$ is achieved provided X_0 is odd and multiplier a = 3+8k or a = 5+8k , k = 0,1,2,…
3. For m a prime number and c = 0, period p = m-1 is achieved provided a has the property that the smallest integer is such that a^k-1 is divisible by m is k = m-1.

Results: (Program printout with output / Document printout as per the format)

Code:

```
#include <iostream>
#include <vector>
#include <cmath> // Include cmath for pow()

using namespace std;

int main() {
   vector<float> arr;
   int x0, a, c, m;

   cout << "Enter initial seed (x0): ";
   cin >> x0;

   int b = 8;
   m = pow(2, b);
   a = 11;
   c = 0;

   for (int i = 0; i < 100; i++) {
      x0 = (x0 * a + c) % m;
      float r = static_cast<float>(x0) / m;
      arr.push_back(r);
   }
```

```
    cout << "Generated Sequence:" << endl;
    for (int i = 0; i < arr.size(); i++) {
        cout << arr[i] << " ";
    }


    float sumGap = 0.0;
    for (int i = 0; i < arr.size() - 1; i++) {
        sumGap += abs(arr[i + 1] - arr[i]);
    }
    float averageGap = sumGap / (arr.size() - 1);

    cout << "\n\nAverage gap between numbers: " << averageGap << endl;

    int period1 = pow(2, b);
    int period2 = pow(2, (b - 2));
    int period3 = m - 1;

    cout<<"period: "<<period1<<endl;
    cout<<"period: "<<period2<<endl;
    cout<<"period: "<<period3<<endl;

    return 0;
}
```



```
Enter initial seed (x0): 47
Generated Sequence:
0.0195312 0.214844 0.363281 0.996094 0.957031 0.527344 0.800781 0.808594 0.894531 0.839844 0.238281 0.621094
    0.832031 0.152344 0.675781 0.433594 0.769531 0.464844 0.113281 0.246094 0.707031 0.777344 0.550781 0
    .0585938 0.644531 0.0898438 0.988281 0.871094 0.582031 0.402344 0.425781 0.683594 0.519531 0.714844 0
    .863281 0.496094 0.457031 0.0273438 0.300781 0.308594 0.394531 0.339844 0.738281 0.121094 0.332031 0
    .652344 0.175781 0.933594 0.269531 0.964844 0.613281 0.746094 0.207031 0.277344 0.0507812 0.558594 0
    .144531 0.589844 0.488281 0.371094 0.0820312 0.902344 0.925781 0.183594 0.0195312 0.214844 0.363281 0
    .996094 0.957031 0.527344 0.800781 0.808594 0.894531 0.839844 0.238281 0.621094 0.832031 0.152344 0
    .675781 0.433594 0.769531 0.464844 0.113281 0.246094 0.707031 0.777344 0.550781 0.0585938 0.644531 0
    .0898438 0.988281 0.871094 0.582031 0.402344 0.425781 0.683594 0.519531 0.714844 0.863281 0.496094

Average gap between numbers: 0.316209
period: 256
period: 64
period: 255
```

Output:

Questions:

1. List down a few real life applications using random numbers as input.

2. List the methods for generating random numbers.

Ans:

1))

1. Simulation and Gaming: Random numbers are commonly used in simulations and gaming to introduce variability and unpredictability. This could be in the form of random events, outcomes, or scenarios.

2. Cryptographic Systems: Random numbers play a crucial role in cryptographic systems for generating keys and ensuring the security of communication.

3. Monte Carlo Methods: Random numbers are extensively used in Monte Carlo simulations for solving mathematical problems, optimizing processes, and making predictions.

4. Randomized Algorithms: Some algorithms use random numbers to introduce an element of randomness and achieve more efficient or faster results. Examples include quicksort with a random pivot and the randomized algorithm for matrix multiplication.

5. Random Sampling in Surveys and Studies: In surveys and scientific studies, random numbers are used to select random samples from a population, ensuring the results are representative.

2))

1. Pseudo-random Number Generators (PRNG):
   - These algorithms generate sequences of numbers that appear random but are actually determined by an initial value called a seed.
   - Examples include the linear congruential generator and the Mersenne Twister.

2. True Random Number Generators (TRNG):
   - TRNGs generate numbers from physical processes that are genuinely random, such as atmospheric noise or electronic noise.
   - Hardware-based TRNGs are often preferred for applications where true randomness is critical, like in cryptographic systems.

It's important to note that while PRNGs are suitable for many applications, they are deterministic and repeat their sequence after a certain point, making them unsuitable for applications where unpredictability and true randomness are essential, such as cryptographic key generation.

Outcomes:

CO1: Apply the experimental process of simulation for model building using simulation languages and tool.
CO2: Generate pseudorandom numbers and perform empirical tests to measure the quality of a pseudo random number generator.

Conclusion: (Conclusion to be based on outcomes)

Thus we successfully implemented Random Number Generator

Grade: AA / AB / BB / BC / CC / CD /D

Signature of faculty in-charge with date

References:

Books/ Journals/ Websites:

Text Book:

Banks J., Carson J. S., Nelson B. L., and Nicol D. M., “Discrete Event System Simulation”, 3rd edition, Pearson Education, 2001.

Websites:

[1] http://en.wikipedia.org/wiki/Pseudorandom_number_generator
[2] http://en.wikipedia.org/wiki/Linear_congruential_generator

.