ISTANBUL KÜLTÜR UNIVERSITY

NETWORK SECURITY
PSEUDO RANDOM NUMBER GENERATOR

Submitted By

ALEYNA ERKİŞİ 1600002059

Project Advisor
Dr. ÖZNUR ŞENGEL

Department of Computer Engineering
İstanbul Kültür University

2021 SPRING

<May>
<2021>

ABSTRACT

When I think about what a random number is I can't think of anything. Because when we say 'random', it doesn't represent anything, alone. For a number to be random, there has to be a list of numbers, so that we can talk about randomness. And, what does random means, is all about mathematics. There are basically two conditions have to be met for a list of numbers to be random. First one; the numbers should be uniformly distributed over a defined interval or set, and, second one; it should be impossible to predict future values based on past or present ones. Some researches say even humans can't think of random numbers really well. Of course, I have to talk about why do we need these random numbers. Random numbers are important in statistical analysis and probability theory. That is why we use computers to do the job for us. Computers can generate truly random numbers, with the help of an external input which we assume to be random. In this project, I used the mouse as the external input and generated so to speak random numbers, which are actually called pseudo random numbers.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1. Problem Statement

Implementing a pseudo random number generator that can be used for keys, IVs, and other cryptographic parameters.

1.2. Project Purpose

Generating pseudo random numbers are important for doing experiments. Computers can generate truly random numbers, but they would need one or more external input which we assume to be random. They inherit the randomness of the world. Machines are deterministic. Their operation is predictable and repeatable, and we need them to be like this. We would like to have the control of the random numbers so that the experiment can be repeated.

This is where pseudo random numbers enter the picture. Random number generators that do not rely on real world phenomena to produce their streams are referred to as pseudo random number generators.

1.3. Project Scope

In this project, I implemented a pseudo random number generator that is inspired from the Fibonacci Series. This generator uses seeds taken from mouse click time and position at an interval of time, with the help of Task Manager in Windows OS.

2. RELATED WORK

In the book [1] The Art of Computer Programming is written by Donald E. Knuth from Stanford University. In the book Knuth really approaches the Programming as art, giving lots of anecdotes from the popular people of the computer world. In the book, it starts with introducing Random Numbers. Examples of usage of the random numbers are given with examples. Also, multiple algorithms are analyzed in detail and given to the reader with experimental tests. One of the given algorithms is Linear Congruential Method, which is one of the simplest but common algorithm in the pseudo random number generator's history.

In the senior thesis [2] written by David DiCarlo from Liberty University, it is talked about the Types and Techniques of the random numbers. There is great real life examples given, such as the case study about the random number generation in Linux OS. It is so interesting because, in the case study it is mentioned that this random number generator is not safe. It lacks forward security because it is only pseudo numbers. There is also a case study about RSA Key Generation. It is said: "Encryption means nothing at all if the random number generators it relies on are poorly constructed." In the thesis.

3. METHODOLOGY

I will now give information how I implemented the project.

3.1. REQUIREMENT ANALYSIS

3.3.1 Textual Requirements / Use Case Diagram

- ✓ Since the only way to use this generator is just calling the function of it, there is no need for a use case diagram.

3.2. DESIGN

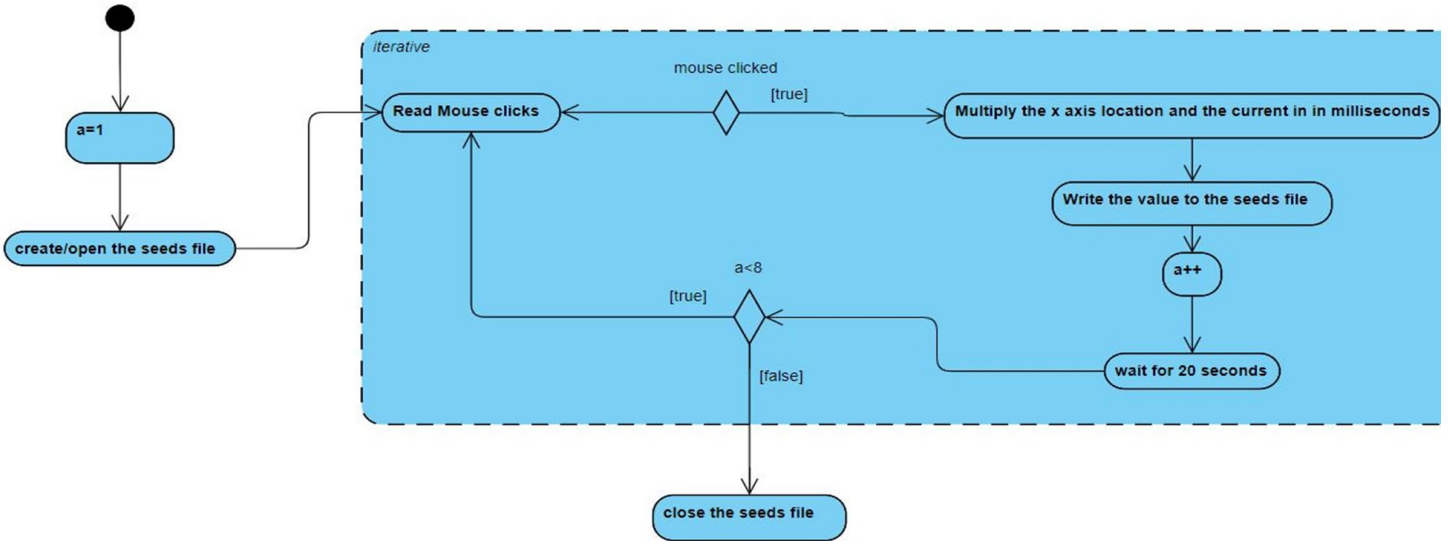

Figure 3.1 : Activity Diagram of the Mouse Detection Program

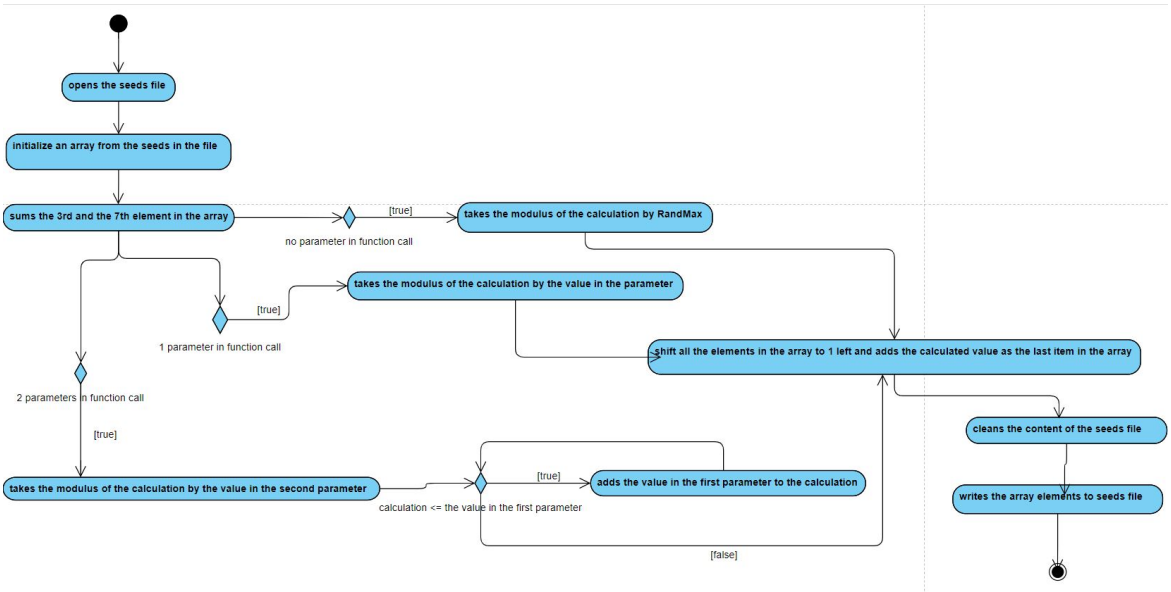

Figure 3.2 : Activity Diagram of Pseudo Random Number Generator

3.3. IMPLEMENTATION

3.3.1 Implementation of Mouse Detection Program

- ✓ I used the win32api library in Python, that has functions to read peripherals' data.
- ✓ Each time the mouse left button is clicked, I read the time in milliseconds and the position of the mouse where it is clicked. For example, my screen resolution is 1920x1080 which means if I click on my screen with my mouse, I can read the position value at most (1920, 1080) as the position point.
- ✓ I have taken the X- coordinate of this point and multiplied it by the time in milliseconds to create a pseudo random number seed value.
- ✓ For the pseudo random number generator to work, I only need 7 seeds. So, the program works until it generates 7 seeds, which means 7 mouse clicks.
- ✓ For the sake of program, to make it more efficient I put an interval time, 20 seconds, between each click. So that the numbers can be as random as possible. Of course, this 20 second is minimum interval.
- ✓ Each time the seed generated I write them to a txt file, soon to be used in the pseudo random number generator.

3.3.2 Implementation of Pseudo Random Number Generator Program

- ✓ The program consists 4 main parts: Reading seeds, generating the random number, changing the seeds, and lastly changing the seeds txt file.
- ✓ First, I opened the seeds txt file created in the mouse detection program and read all the data in it and save them in an array.
- ✓ I generated the random numbers with the calculation below:

$$S_n \equiv S_{n-j} \star S_{n-k} \pmod{m}, 0 < j < k$$

Figure 3.3 : Lagged Fibonacci Method

- ✓ I selected j as 3 and k as 7; and the ★ as summation. I wanted to make the program's flexibility better, thus *m* can be either selected by the user by putting a parameter whilst calling the function, so they can select a range, or it is selected as 32767. The user can select a minimum value as well as selecting a maximum one whilst in the function calling.
- ✓ After the generation of the random number, I changed the seeds array as below:

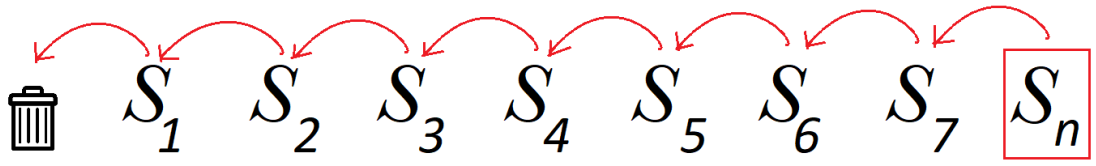

Figure 3.4 : Lagged Fibonacci Method Shifting

- ✓ Lastly, I wrote the new seeds array into the seeds txt file.

3.3.2 Implementation of Mouse Detection Program Task

I created a task in the task scheduler in Windows, so that every time the computer boots it will start. That way it will keep renewed by creating new seeds for the user, and this will increase the randomness. First, I created a batch file from the python program. Then I created the task in task manager. I used the below specifications to do so:

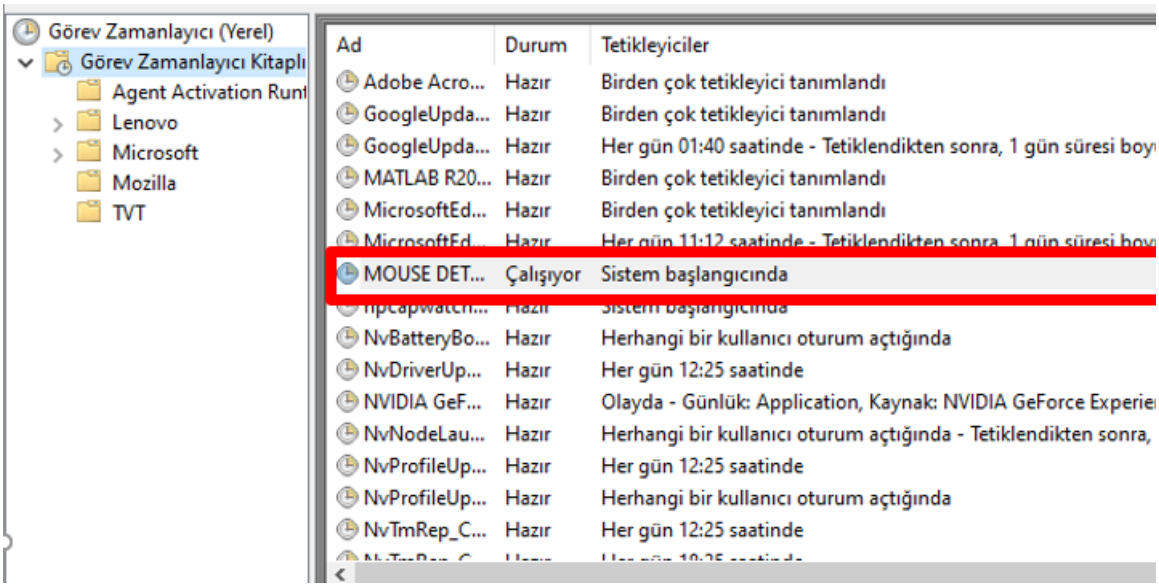

Figure 3.5 : Mouse Detection Task Manager

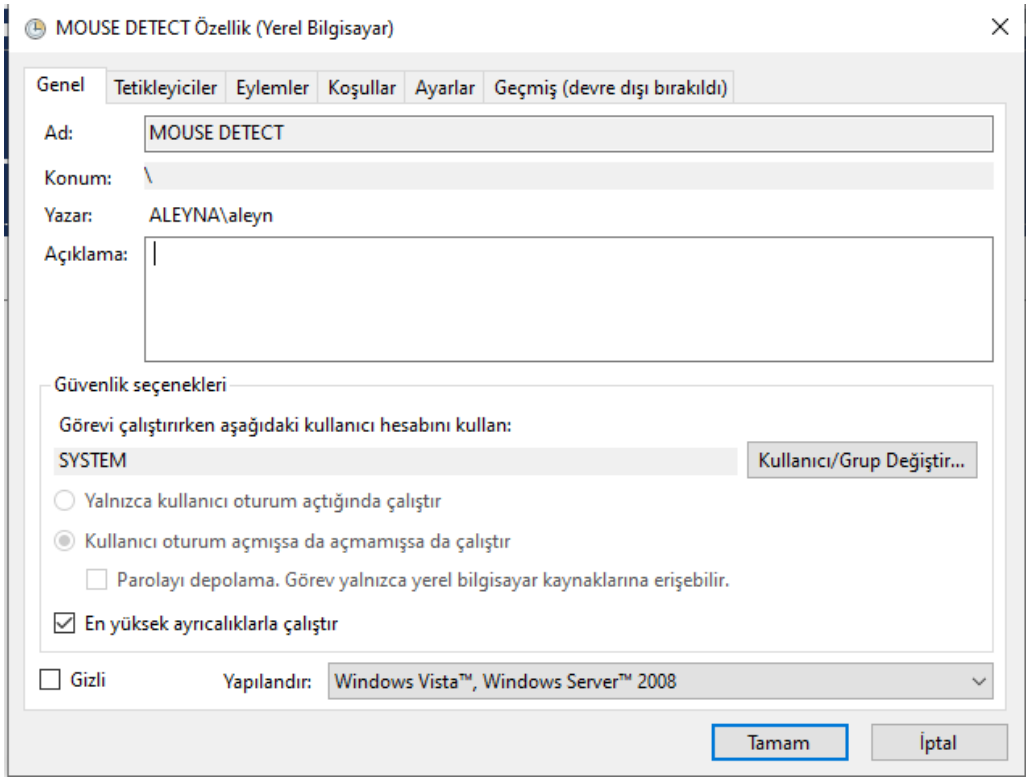

Figure 3.6 : Mouse Detection Task General Settings

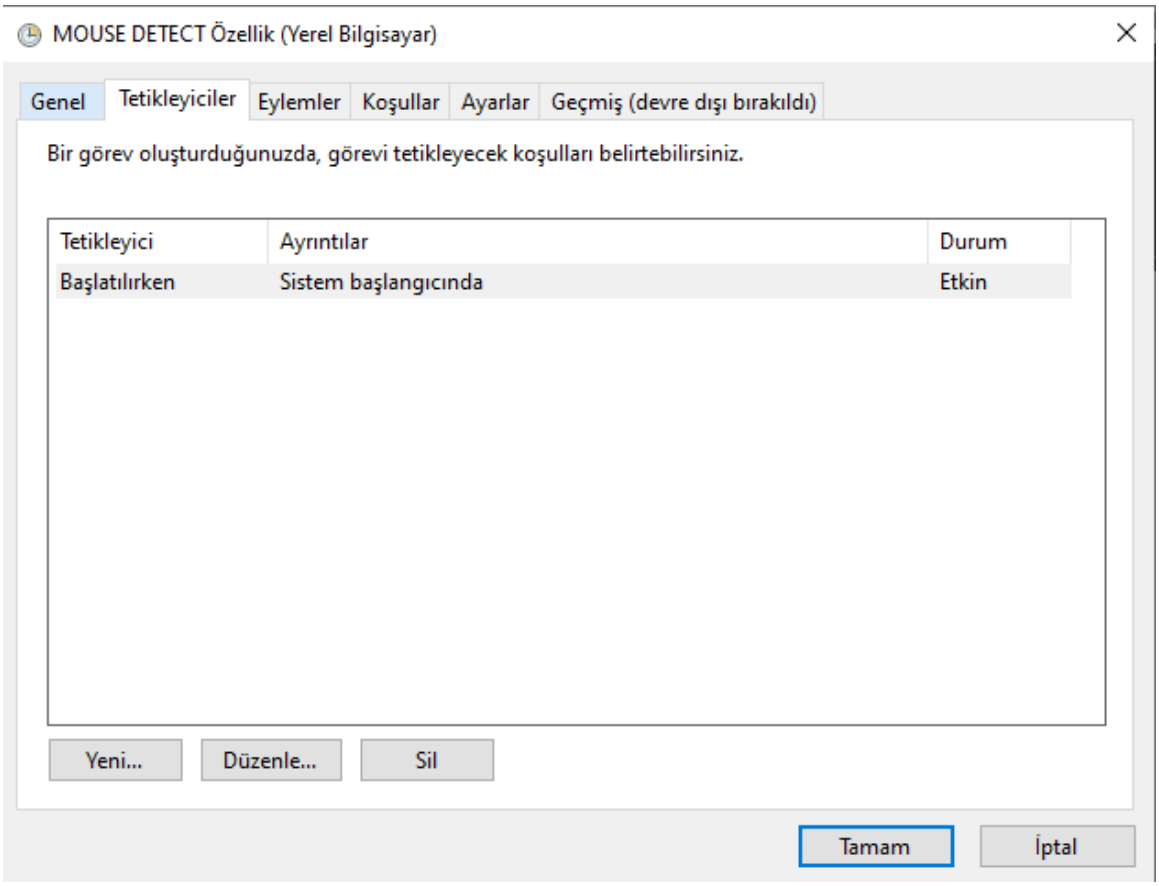

Figure 3.7 : Mouse Detection Task Trigger Settings

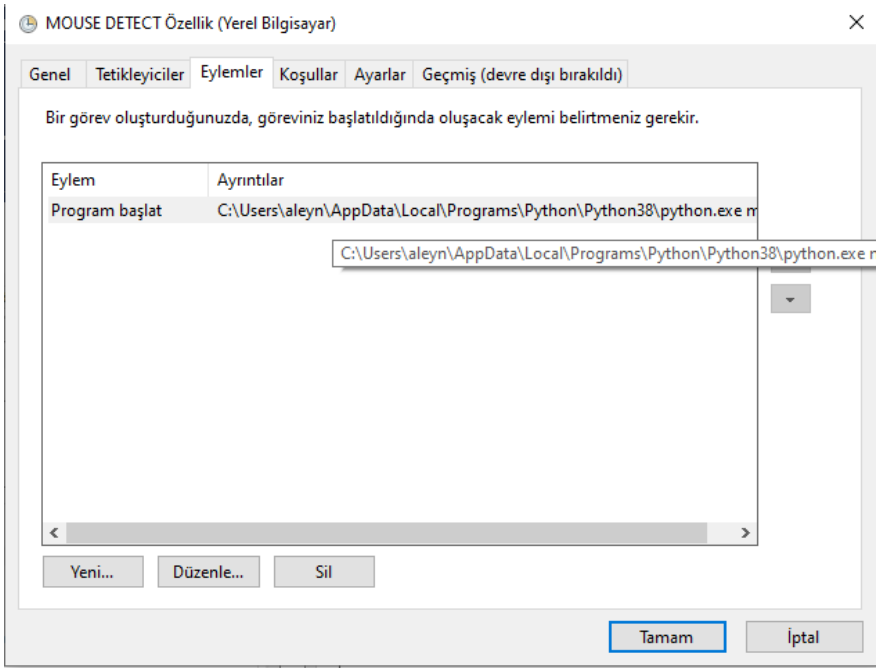

Figure 3.8 : Mouse Detection Task Action Settings

3.4. TESTING

- ✓ As for testing, I used the test of randomness evaluated by the Z-statistics.[3] A z-score describes the position of a raw score in terms of its distance from the mean, when measured in standard deviation units. The z-score is positive if the value lies above the mean, and negative if it lies below the mean. You can see the test results below:

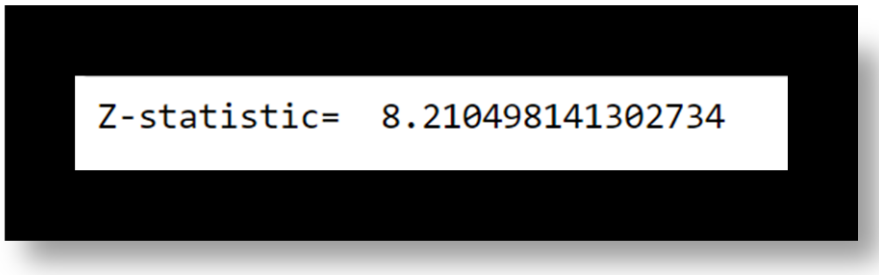

Figure 3.9 : Randomness Test with Z-statistics

- ✓ I also tested the random numbers, by the online Randomness of Statistical Sampling: The Runs Test from University of Baltimore implemented by Professor Hossein Arsham.[4] You can see the test results below:

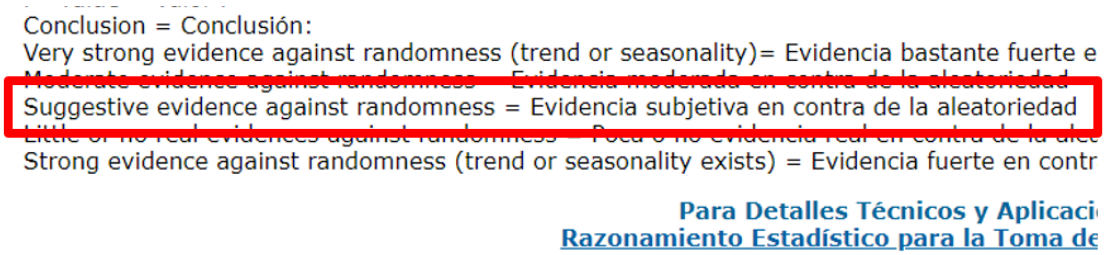

Figure 3.10 : Randomness Test with The Run Test from the Baltimore University

4. DISCUSSION

In this project, I implemented a pseudo random number generator that is inspired from the Fibonacci Series. This generator uses seeds taken from mouse click time and position at an interval of time, with the help of Task Manager in Windows OS.

The importance of the pseudo random numbers is high when it comes to computer security. From the pseudo random numbers, we can generate keys for encryption algorithms, [5] which then used for decryption purposes. So, if the random number generated by our algorithm is insufficient in randomness, meaning, if someone can guess that random number, then they can decrypt the cipher without authorization. At the core of random number generators lay the seed numbers. These seeds can be selected from an external hardware. Because these external hardware inherit the randomness of the nature. One cannot tell the radioactivity level in an area just by guessing it. Or one cannot guess the user's mouse clicks when it is done, where it is done. Of course, under some circumstances this can be manipulated by some attackers. It is as safe as any program can be, which is not 100% for any program made until now. But what I can do, as a programmer, while implementing the program, is to decrease this vulnerability. And, I have done it by using an external hardware to implement my seeds.

CONCLUSIONS

In conclusion, implementing truly random numbers are impossible with the help of computer, but pseudo random numbers can be implemented instead. Pseudo random numbers can be use in encryption algorithms, simulations, decision making etc. Their importance is real in computer context and in nature itself. They are good for doing tests because they can be created again, thanks to computers. That is why, even today, new algorithms are developed to have a better randomness.

REFERENCES

[1] Donald E. Knuth(1997), The art of Computer Programming, Stanford University URL: https://doc.lagout.org/science/0_Computer%20Science/2_Algorithms/The%20Art%20of%20Computer%20Programming%20%28vol.%202_%20Seminumerical%20Algorithms%29%20%283rd%20ed.%29%20%5BKnuth%201997-11-14%5D.pdf

[2] David DiCarlo(2012), Random Number Generation: Types and Techniques , Liberty University https://digitalcommons.liberty.edu/cgi/viewcontent.cgi?article=1311&context=honors

[3] GeeksforGeeks. (2020, June 8). Runs Test of Randomness in Python. https://www.geeksforgeeks.org/runs-test-of-randomness-in-python/

[4] Arsham, P. H. A. (2015). Test for Randomness [Slides]. Http://Www.Ubalt.Edu/. https://home.ubalt.edu/ntsbarsh/business-stat/otherapplets/Randomness.htm

[5] For the Love of Computing: The Lagged Fibonacci Generator, Where Nature Meet Random Numbers by Prof Bill Buchanan, Medium. https://medium.com/asecuritysite-when-bob-met-alice/for-the-love-of-computing-the-lagged-fibonacci-generator-where-nature-meet-random-numbers-f9fb5bd6c237

APPENDIX

```
// MOUSE DETECTION (PYTHON)

import win32api
import time
import pyautogui
a=1
f = open("seeds.txt", "a")
while a<8:
    check= win32api.GetKeyState(0x01)
    if check < 0:
        print("left button is clicked")
        mtime= round(time.time())
        x, y = pyautogui.position()
        final=mtime*int(str(x).rjust(4))
        f.write(str(final))
        f.write("\n")
        a+=1
        time.sleep(20)
        f.close()
```

```
// RANDOM NUMBER GENERATOR (C++)

#include <iostream>
#include <fstream>
#include <string>
using namespace std;

       unsigned long long int seeds[7]; //initial seeds (they are all positive
anyways, good to have a bigger range data type)
       long long int randomNum; //positive or negative random number
       int randMax = 32767;

       //random number between 0 and randMax
       long long int randomNumber();
       //random number with ending range ( 0 < randomNumber < range)
       long long int randomNumber(int range);
       //random number with starting and ending range ( rangeI < randomNumber <
randomL)
       long long int randomNumber(int rangeI, int rangeL);
       //opens the seed file and fill the seed array with the numbers from the
file
       void seedArray();
       //deletes the first element in the array, shift all elements to one left
index and adds the newly created random number to the last index
       void changeArray();
       //adds the new seeds array to the seeds file to use it for the next time
the program launches
       void fileChanger();
       //random  number generator with no parameter
       long long int Random();
       //random number gneerator with 1 parameter
       long long int Random(int range);
       //Random number generator with 2 parameter
       long long int Random(int rangeI,int rangeL);
```

```
	void seedArray()
	{
		// Read from the seed text file
		ifstream mySeedsFile("Seeds.txt");
		int i = 0;
		string myText;

		// Use a while loop together with the getline() function to read the
file line by line
		while (getline(mySeedsFile, myText)) {
			// Output the text from the file
			seeds[i++] = stoull(myText); //convert into unsigned long
long int number
		}

		//close the file
		mySeedsFile.close();

	}

	long long int randomNumber()
	{
		randomNum = ((seeds[2]%randMax) + (seeds[6]%randMax)) %randMax;

		return randomNum;
	}

	long long int randomNumber(int range)
	{
		randomNum = ((seeds[2] % range) + (seeds[6] % range)) % range;
		return randomNum;
	}

	long long int randomNumber(int rangeI, int rangeL)
	{
		randomNum = ((seeds[2] % rangeL) + (seeds[6] % rangeL)) % rangeL;
		while (randomNum <= rangeI)
			randomNum += rangeI;

		return randomNum;

	}

	void changeArray()
	{
		for (size_t i = 0; i < 6; i++)
		{
			seeds[i] = seeds[i + 1]; //shift to left by one (all elements
except the last one since there is no 8th element(aka index 7)
		}
		seeds[6] = randomNum; //puts the newly created random number to the
seeds array as the last item
```

```
}

void fileChanger()
{
	ofstream mySeedsFile;

	//first clean the file content
	mySeedsFile.open("C:/Users/aleyn/Desktop/Seeds.txt");
	mySeedsFile.close();

	//open it again this time put the new seed array into the file
	mySeedsFile.open("C:/Users/aleyn/Desktop/Seeds.txt");

	for (size_t i = 0; i < 7; i++)
	{
		mySeedsFile << seeds[i] << endl;

	}

	mySeedsFile.close();

}

//functions to be used in main

long long int Random()
{
	//starts with opening the seed file and initializing the seed array
	seedArray();
	//generates the random number
	randomNumber();
	//changes the array with the newly generated random number
	changeArray();
	//changes the seed file with the new seed array
	fileChanger();

	return randomNum;

}


long long int Random(int range)
{
	//starts with opening the seed file and initializing the seed array
	seedArray();
	//generates the random number
	randomNumber(range);
	//changes the array with the newly generated random number
	changeArray();
	//changes the seed file with the new seed array
	fileChanger();

	return randomNum;
}

long long int Random(int rangeI, int rangeL)
{
	//starts with opening the seed file and initializing the seed array
```

```
                seedArray();
                //generates the random number
                randomNumber(rangeI,rangeL);
                //changes the array with the newly generated random number
                changeArray();
                //changes the seed file with the new seed array
                fileChanger();

                return randomNum;
        }
```

```
// DEMO (C++)

int main()
{


        for (size_t i = 0; i < 10; i++)
        {
                cout << Random() << endl;

        }

        cout <<endl<< "1 parameter " << endl;
        for (size_t i = 0; i < 10; i++)
        {
                cout << Random(100) << endl;

        }

        cout <<endl<< "2 parameters " << endl;
        for (size_t i = 0; i < 10; i++)
        {
                cout << Random(50,100)<< endl;

        }

}
```

```
// BATCH FILE

"...\AppData\Local\Programs\Python\Python38\python.exe" "...
\mouseDetectionTask.py"
pause
```