BIL105E – Introduction to Scientific and Engineering Computing (C)

Spring 2015 - 2016

CRN: 21837

Homework 2

Assignment Subject: ***Writing a C program which produces random numbers with two pseudo-random number generator (PRNG) functions which the user can not predict it beforehand***

Assignment Due Date: ***11.04.2016 23:00***

Submission Date: ***11.04.2016***

Student Name: ***Korel Chairoula***

Student ID: ***150140910***

e-mail: ***korel_hayrullah@hotmail.com***

Introduction Section

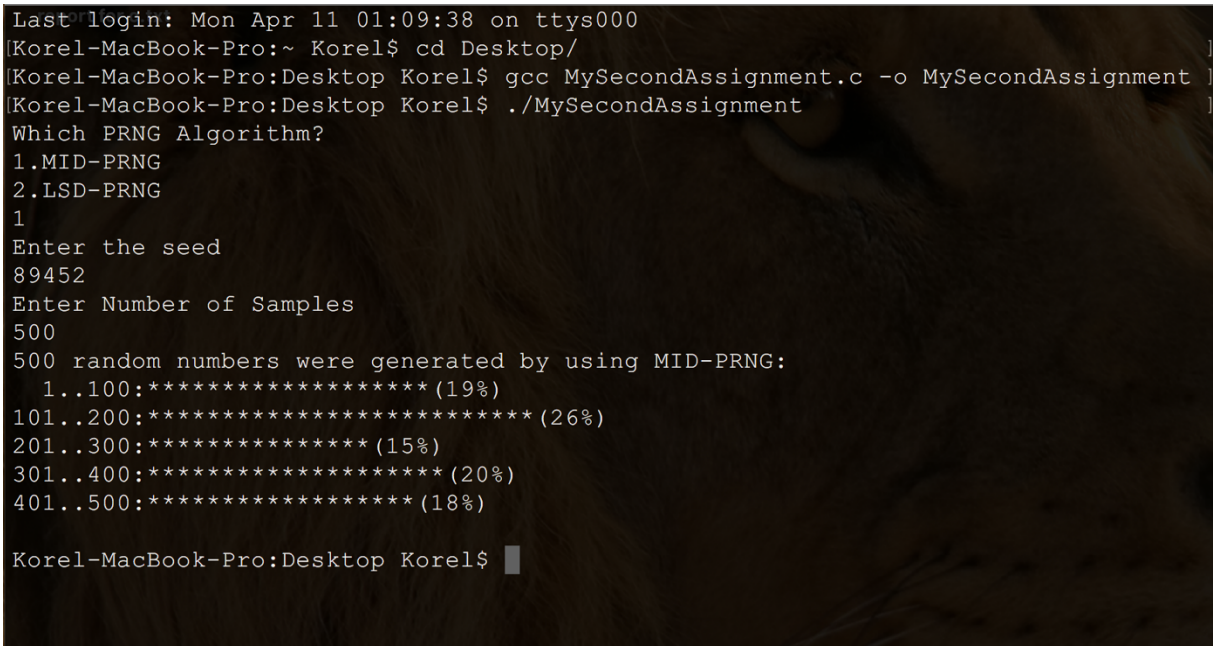
First of all, working on this project was also fun like the first project. The project was about writing two pseudo-random number generator (PRNG) functions which the user can not predict it beforehand and printing the histogram of the generated random numbers. There are two PRNG functions which are MID-PRNG and LSD-PRNG functions. The MID-PRNG function takes a 5-digit seed and number of samples which the user defines here how many random number the user wants to generate. Then the program takes the seed takes the square of the seed and gets the 5 middle digits as random number. The program does this until number of samples entered and every time it gets the seed from the 5 middle digits of the square. So, the user enters just one time the seed. Also, it checks how to take out the 5 digits from the square and this is x12345x+1 for even numbers and x12345x for odd numbers which the x is the numbers which will be taken out. On the other hand, the LSD-PRNG function has the same logic, but it only multiplies the number with 73 and takes the 5 digits from left to right. However, the difference in LSD-PRNG functions is that it checks the the seed entered from the user if it is even or ends with a 5. If so, the user has to try again. After the execution of these PRNG functions the program takes the generated random numbers and sorts them to appropriate ranges. For every generated random number, the program takes mod 500 of the number and adds 1 to it. After that, it determines in which range the number will be which there are 5 ranges starting from 1 to 100, 101 to 200, 201 to 300, 301 to 400 and ending with 401 to 500. On the final stage, it takes the ranges and prints the histogram for every range which tells the user for every generated random number in which range the number is with a percentage. The goal is to create random numbers with the seed entered from the user and can not be predicted beforehand.

Development Environment

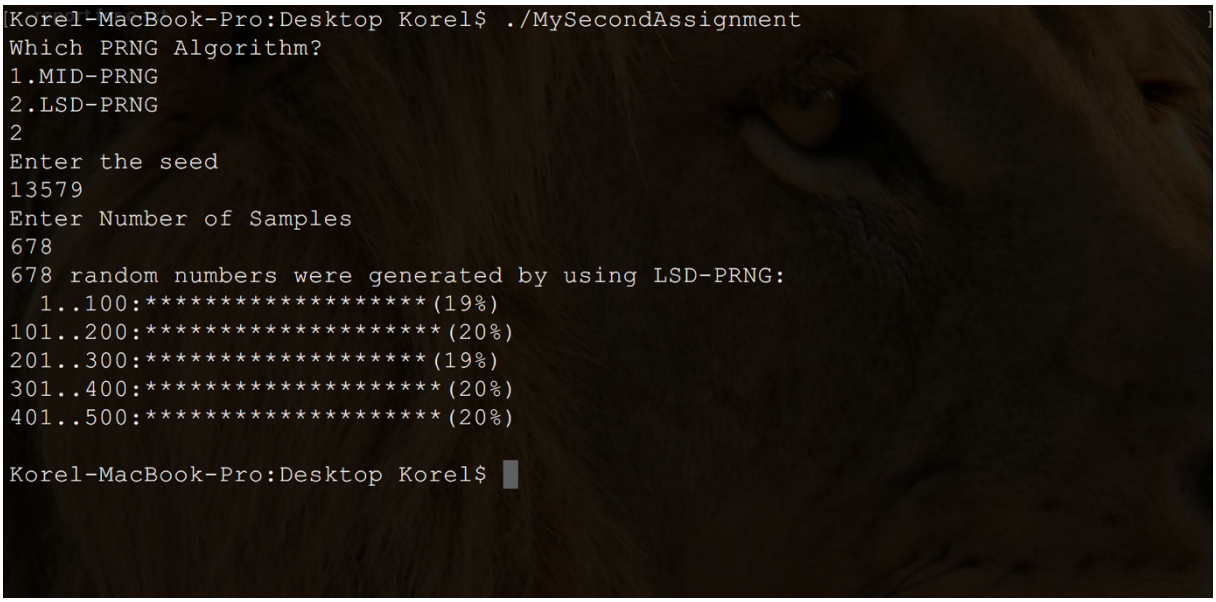
Operating System: ***OSX El Capitan***
Programming Language: ***C***
Compiler: ***GCC***
Source File*: My program includes one file c named* ***150140910.c***
Header Files: *The header file used is* ***<stdio.h>***

I used header files <stdio.h> because it is the default library in order to run default functions. After finishing coding my program I used terminal on my MacBook to compile it via gcc. My command was “Korel-MacBook-Pro:~ Korel.Hayrullah$ gcc 150140910.c”. In order to run it I used in terminal “./a.out” to run my program. There is a easier way to compile and simultaneously change the name is “gcc 150140910.c –o MySecondAssignmentC” and then again to run it “./MySecondAssignmentC”. Here are some screenshots to explain this via image and in the meantime my program after run.

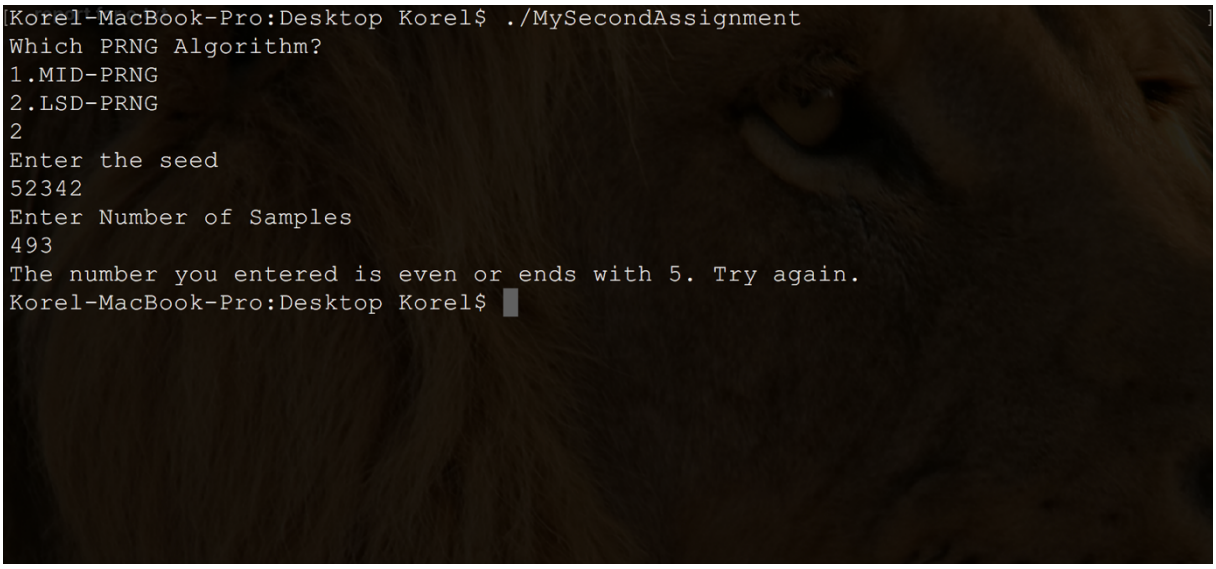
An example of the program using MID-PRNG function and compiling process.

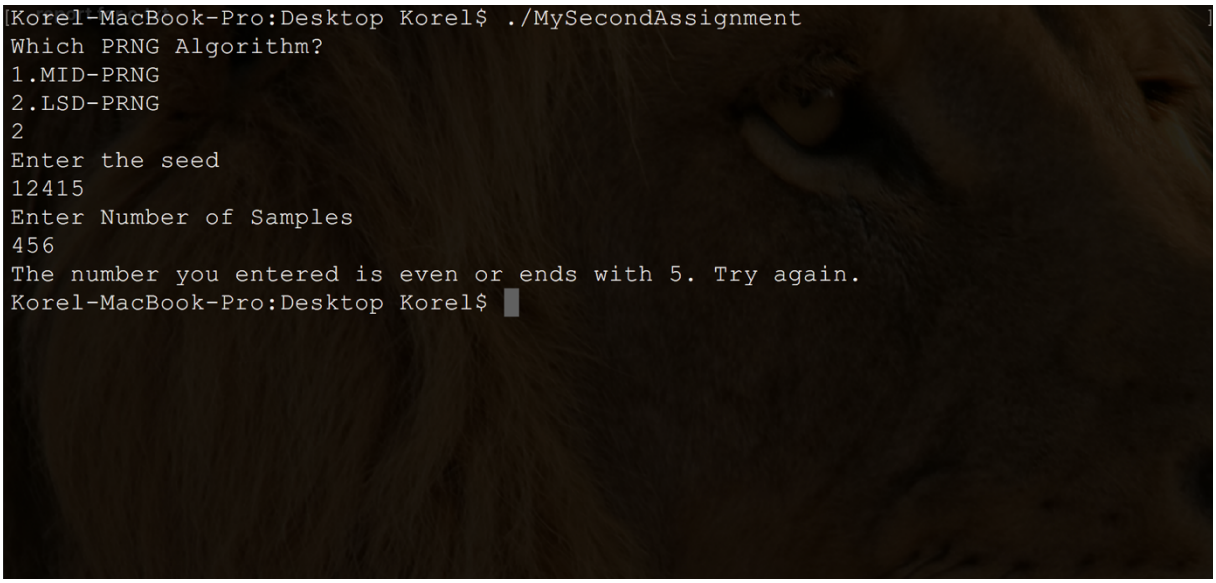


An example of the program using LSD-PRNG function.



More examples of the LSD-PRNG function which warns the user that the number is even or ends with a 5.





Data Structure and Variables

In this project I used **printf** and **scanf** functions, **for** loops and **if** statements. Also, in total there are 4 functions with the type **void**. These are: **void MID_RNG(long *number)** in which takes the seed entered from the user and takes its square and takes out the 5 middle digits (for example x12345x+1 for even digits and x12345x for odd digits which x here is the numbers taken out), **void LSD_RNG(long *number)** in which takes the seed entered from the user and multiplies it by 73 and takes the 5 digits from left to right, **void take_samples(long *number)** in which takes every random generated number takes it mod 500 and adds 1 and determines in which of the five ranges it will and increments the range

in by one and **void print_histogram(int range1_100,int range101_200, int range201_300, int range301_400, int range401_500)** in which takes the ranges previously calculated and incremented from the **take_samples** function and prints the histogram and percentage for all generated random numbers . The void in this type of writing means that it does not return anything. The ***number** in the functions **MID-RNG** and **LSD-PRNG** is for pointer. While calling and sending something to these functions you have to use **"&"** and the types must match. For example: if the type of the value is **"int"** you have to tell the function that the type is **"int"** too and calling must be like **"MID_RNG (&seed);".** This is called call by reference. Lastly in **print_histogram** function I used **"%c"** to print the **"%"** sign because I got "incomplete format specifier" warning and to fix this I made it like (**printf(%c,'%');**). Here are the variables which I used in this project:

Range variables which all have the same operation defined global in order to store the numbers and not to take any wrong number which are incremented in the **take_samples** function and then sent to the **print_histogram** function:

> **int range1_100 = 0; //** stands for the range 1 to 100 which the program will increment in that range after taking the mod 500 and adding 1 to every produced number.
>
> **int range 101_200 = 0; //** stands for the range 101 to 200 which the program will increment in that range after taking the mod 500 and adding 1 to every produced number.
>
> **int range 201_300 = 0; //** stands for the range 201 to 300 which the program will increment in that range after taking the mod 500 and adding 1 to every produced number.
>
> **int range 301_400 = 0; //** stands for the range 301 to 400 which the program will increment in that range after taking the mod 500 and adding 1 to every produced number.
>
> **int range 401_500 = 0; //** stands for the range 401 to 500 which the program will increment in that range after taking the mod 500 and adding 1 to every produced number.

int control; // stands for controlling whether it will be 1 or 2. For example 1 for MID-PRNG function and 2 for LSD-PRNG function.

int i; // stands for the for loop in order to control it. In my program it is equalized every time in the for loop to 0. This variable is defined global in order not to define it again and again.

int num_samples; // at the very beginning the user enters the number of samples and this stands for how many random numbers will be generated.

long seed; // at the very beginning the user enters the seed which in this program the user should enter 5 digits. After that, the seed is sent to the chosen functions to produce random

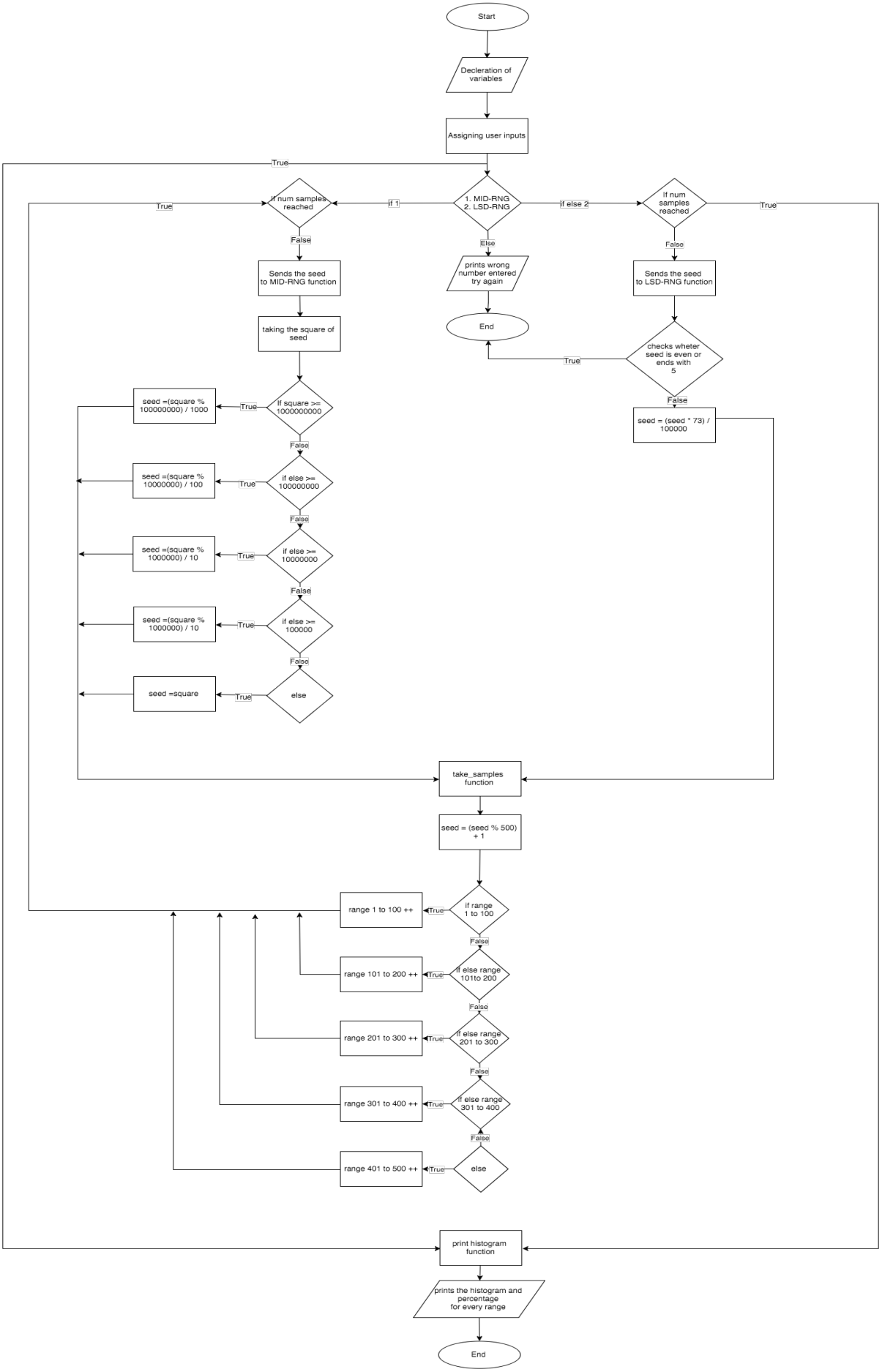
numbers. It is entered once by the user and in the second time is takes the processed number as seed. The long seed is scanned with “%ld”.

long square; // in the MID-RNG function it takes the it takes the seed which is sent “&seed” from main and retrieved in the MID-RNG function “*number” which stands for pointer. The long square variable multiplies the *number with *number and holds it for the coming calculations.

int sample; // sample here is for the processed number every time both for MID-RNG and LSD-RNG functions which every time the processed the number is sent like seed and the number mod 500 is taken and added with 1. After that, the calculated number is hold in that variable sample and checked for which range the processed number should go and incremented.

Program Flow

Because of the globally defined ranges I draw an arrow between “Assigning user inputs” and “1.MID-RNG 2.LSD-RNG”. Every generated number after process is incremented and saved. After that they are taken to the print_histogram function which prints the histogram and percentage.


Start
Decleration of variables
Assigning user inputs
1. MID-RNG
2. LSD-RNG
if 1
if else 2
Else
prints wrong number entered try again
End
If num samples reached
True
False
Sends the seed to MID-RNG function
taking the square of seed
If square >= 1000000000
seed =(square % 100000000) / 1000
If else >= 100000000
seed =(square % 10000000) / 100
if else >= 10000000
seed =(square % 1000000) / 10
If else >= 100000
seed =(square % 1000000) / 10
else
seed =square
If num samples reached
Sends the seed to LSD-RNG function
checks wheter seed is even or ends with 5
seed = (seed * 73) / 100000
take_samples function
seed = (seed % 500) + 1
if range 1 to 100
range 1 to 100 ++
If else range 101to 200
range 101 to 200 ++
If else range 201 to 300
range 201 to 300 ++
If else range 301 to 400
range 301 to 400 ++
else
range 401 to 500 ++
print histogram function
prints the histogram and percentage for every range
End

Conclusion

In conclusion, I enjoyed working on this project because every time I compiled the program I got errors, warnings and sometimes I did not get nothing, but the program did not work as expected. All these struggles took a lot of time to understand what is missing or what is the error. You had to check every possible number that can make the program work not as expected. For example, in the MID-RNG function takes the seed and takes it square. After that it takes the middle 5 digits which every time you need to calculate it well how the program should take these 5 digits. Also, sometimes the square of the number could came like 1403459043 which in this case the 5 middle digits are 03459. However, like in normal life it ignores the 0 and the number becomes 4 digits 3459. You had to check also this case which at the very beginning I could not think of that. The rest of it also need to think of and calculate. Like in the first project this project also had a good impact on learning, practicing and understanding the working principle and logic of programming language C. All in all, the project contributed to understand how it works and responses.