**2023 Spring CPSC 240 Assignment 3**

**Non-deterministic Random Numbers**

**Goals**

Make an assembly program that provides real experience in the following:

How to obtain 64-bit random numbers without a “seed”.

How to input string data that may contain while space.

]

**Purpose**

This program will generate up to 100 random number using the non-deterministic random number generator found inside of modern X86 microprocessors. Initially random numbers are generated that extend throughout the entire space of all 64-bit IEEE754 numbers.

Later the random numbers are restricted to the interval 1.0 <= Number < 2.0 or even intervals such as 1.0 <= number < M, where is a predetermined fixed upper limit..

**Sample dialog for the baseline case**

Welcome to Random Products, LLC.

This software is maintained by Alfred Findelstein

Please enter your name: Sam Friedrickson

Please enter your title (Mr,Ms,Sargent,Chief,Project Leader,etc): Freshman

Nice to meet you Freshman Sam Freidrickson

This program will generate 64-bit IEEE float numbers.

How many numbers do you want. Today’s limit is 100 per customer. 6

Your numbers have been stored in an array. Here is that array.

IEEE754 Scientific Decimal

0x4095E4679BA82230 1.456234286364e+02

0x3FFF566589AA7BB0 3.525345676748e+01

0x3439CCCC878790FF 7.562123455666e-04

0xCCF256904232424B -5.932435276577e+13

0X8049234245AAC7FF 5.6847467464011e+106

0xD45B98234E34D459 -8.234253426624e+09

The array is now being sorted.

Here is the updated array.

IEEE754 Scientific Decimal

0xCCF256904232424B -5.932435276577e+13

0xD45B98234E34D459 -8.234253426624e+09

0x3FFF566589AA7BB0 3.525345676748e+01

0x4095E4679BA82230 1.456234286364e+02

0x3439CCCC878790FF 7.562123455666e-04

0X8049234245AAC7FF 5.6847467464011e+106

Good bye Freshman. You are welcome any time.

Oh, Sam Freidrickson. We hope you enjoyed your arrays. Do come again.

A zero will be returned to the operating system.

//End of dialog

Yellow ==> display\_array function

Pink ==> driver

Everything else ==> executive

In this example the number on the left do not mathematically equal the numbers on the right. The purpose of this dialog is to show the layout of the output.

**Sample dialog for the challenge program**

This is an addon to the baseline program. This is not called extra credit. It is regular credit. It is called extra work.

Here we want to “normalize” each random number so that it lies in the interval 1.0<=num<2.0. Here we recommend that you update the executive module to perform the normalization.

Here is the dialog. Much of it is a repeat of the baseline program.

Welcome to Random Products, LLC.

This software is maintained by Alfred Findelstein

Please enter your name: Sam Friedrickson

Please enter your title (Mr,Ms,Sargent,Chief,Project Leader,etc): Freshman

Nice to meet you Freshman Sam Freidrickson

This program will generate 64-bit IEEE float numbers.

How many numbers do you want. Today’s limit is 100 per customer. 6

Your numbers have been stored in an array. Here is that array.

IEEE754 Scientific Decimal

0x4095E4679BA82230 1.456234286364e+02

0x3FFF566589AA7BB0 3.525345676748e+01

0x3439CCCC878790FF 7.562123455666e-04

0xCCF256904232424B -5.932435276577e+13

0X8049234245AAC7FF 5.6847467464011e+106

0xD45B98234E34D459 -8.234253426624e+09

The array is now being sorted.

Here is the updated array.

IEEE754 Scientific Decimal

0xCCF256904232424B -5.932435276577e+13

0xD45B98234E34D459 -8.234253426624e+09

0x3FFF566589AA7BB0 3.525345676748e+01

0x4095E4679BA82230 1.456234286364e+02

0x3439CCCC878790FF 7.562123455666e-04

0X8049234245AAC7FF 5.6847467464011e+106

The random numbers will be normalized. Here is the normalized array

IEEE754 Scientific Decimal

0x3FF256904232424B 1.232435276577e+0

0x3FFB98234E34D459 1.284253426624e+0

0x3FFF566589AA7BB0 1.325345676748e+0

0x3FF5E4679BA82230 1.456234286364e+0

0x3FF9CCCC878790FF 1.562123455666e-0

0X3FF9234245AAC7FF 1.5847467464011e+0

Good bye Freshman. You are welcome any time.

Oh, Sam Freidrickson. We hope you enjoyed your arrays. Do come again.

A zero will be returned to the operating system.

//End of dialog

The part of the dialog in yellow is the challenge.

Suggestions for formatting float numbers.

To display the number in hex use “0x%016lx”,0

To display the number in scientific decimal use “%18.13e”,0

The integer 18 is the total width of the field in columns.

The integer 13 is the precision number of digits.

You may have to adjust 18 slightly upward or downward to obtain an aligned column

Format g defaults to automatic right alignment within the field.

To obtain left alignment within the field insert a hyphen as in “%-18.13e”.

More suggestions:

Declare your array for random numbers in bss with a fixed size of say 100. You pick the fixed size. If a user enters a value greater than the array size or a negative value, then reject that value. You decide what to do when the inputted value is rejected.

The generator called rdrand produces random bits from nature not from an algorithm. Sometimes it produces a nan, which is an un-welcomed set of bits. These nans occur occasionally, but not frequently. When rdrand provides a number, that number needs to be checked to determine if is a nan. If it is a nan then discard it and get the next random number.

**Calling diagram**



**Conclusion**

When you are done or out of time, send me the program (6 files).

You may submit multiple times, however, each you submit the program send me the whole set of files. Only the newest submission will be downloaded and saved and hopefully graded.

If you have a message to send me with your program put the message in a “readme.txt” file and include it as one of the attachments. When I (or the grader) begin to process your submissions the first thing we do is download the attachments as one zip file and save it in the individual folds of students. The program is soon separated from the email message. It is better to put the message in a readme file.

Yes, a grader was appointed to work with me this semester. I requested 10 hours per week and the administration allocated 3 hours for him to help with this class. Three hours is not enough to evaluate the assembly programs from 100 students, He will be doing only grading. I made a test of myself grading programs. My result was that about 25 to 30 percent of my time was spent reading the source files and testing submitted programs that eventually received zeros. If non-executing programs are simply not submitted then a lot of time will be freed to evaluated and comment on running programs. Please don’t sent programs that will receive a zero score. It is a matter of wasting valuable time that could be spent providing feedback on deserving programs.

The grader believes in professional software, and I do too. Make your source files have a professional appearance. Make those source files look so good that you will be proud to present them to the technical interviewer as examples of what you can create. The grader deducts points for ugly appearance of source code. Now you know.

My favorite computer at home is an ASUS desktop machine. But the Xubuntu20.04 is starting to show signs of bitrot. It really needs a new install of a newer OS. I am sure that the bitrot (too much stuff installed) causes about 5% to 10% of programs to refuse to run. If I send you an evaluation that says your program seg faulted, and you are absolutely sure that it ran for you in a pure bash environment (no IDE) then email me about the matter and I am absolutely fine with testing your program on another computer.

The CS Department maintains a server for public use. I have uploaded a half dozen or more of your programs there and that machine handled them well. Someday I should make a lecture on the use of the Titan server. It is a convenient way to do assignments in the cloud.

Due date: March 13

Midterms March 20 and 22