**CSE-443/543: High Performance Computing**

**Lab #01**

Max Points: 35

**Objective**: The objective of this exercise is to:

* Build experience with compiling code at the command line and constructing batch shell scripts
* Build C/C++ programming skills
* Gain experience with the use of random numbers for simulation
* Gain experience computing averages and standard errors

Fill in answers to all of the questions. For almost all the questions, you can simply copy-paste appropriate text from the shell/output window into this document or your Google spreadsheet. You may discuss the questions with your instructor and the TA.

# Background

Random numbers are often used in high-performance computing, for computation, simulation and analysis. To be completely honest, "random" numbers are not used frequently but rather "pseudorandom" numbers which are computed by an algorithm but have many of the properties of actual random numbers. Issues with actual random numbers include:

* Difficulty finding sources for them
* Speed of number generation
* True randomness impedes debugging and optimizing programs

Pseudorandom numbers do not suffer from these difficulties, but have the problem that they are not truly random and seemingly small correlations or flaws in pseudorandom numbers can have a significant impact on the results of simulations. If you are interested in this topic, see the paper Monte Carlo simulations: Hidden errors from ‘‘good’’ random number generators (A. M. Ferrenberg, D. P. Landau, and Y. J. Wong Phys. Rev. Lett. 69, 3382 (1992)) included in the Canvas module for today’s session. In addition to talking about random numbers, it also demonstrates the use of statistical errors in a scientific study.

For this lab I am giving you a C++ program that generates a single pseudorandom number in the range 0..1 using the built-in rand\_r() method in the GNU standard library. You should modify this program so that it simulates the roll of three 6-sided dice as demonstrated in class today.

When your program can successfully simulate rolling three dice you should modify it as follows:

1. It should compute the sum of the three dice
2. It should simulate rolling the 3 dice 1000 times, keeping track of the sum in an array.
3. At the end of the run it should write out the number of times each total (from 3 to 18) occurred during the run.

You will repeat this 5 times and record the results in your Google spreadsheet for Lab 01.

To construct the shell script for the experiment, feel free to use the man command or search online for help. Useful commands or concepts include, bash, grep, sort, uniq, and the /proc filesystem. Hint: With the right option you can do steps 3b and 3c below using just the grep command.

# Experiment

1. Download the file Lab01.cpp from Canvas to your workstation.
2. If you didn’t do this in the first session, download the file cpplint.py from Canvas to your workstation.
3. Write a shell script called Lab01.bash that does the following:
   1. Invokes /bin/bash as a login shell.
   2. Prints a single line of “model name” information from /proc/cpuinfo
   3. Prints a single line of “MemTotal” information from /proc/meminfo
   4. Checks the Lab01.cpp code for errors using cpplint.py.
   5. Compiles the program using the Gnu C++ compiler (g++), naming the output file Lab01 and using our standard compiler options:
      1. –O3 --std=c++14 –Wall
   6. Runs the program Lab01
4. When your shell script runs correctly, modify the Lab01.cpp code as requested above to roll three dice instead of one, and do 1000 trials, keeping track of the sum of the dice in an array. At the end of the run your program should write out the array of counts. Make sure to check your code with cpplint.py.
5. When your program is working correctly, run your shell script 5 times and enter your results in the Google spreadsheet for this assignment.

# Apparatus (experimental platform)

The experiments documented in this report were conducted on the following platform (fill in the Details column using information determined in your shell script):

|  |  |
| --- | --- |
| Component | Details |
| CPU Model | Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz |
| Main Memory (RAM) size | 8071544 kB |

# Observations: Counts from simulation of 3 dice

In your Google spreadsheet, record the histograms of 3-dice sums you obtained by running your simulation program.

# Analysis:

Using the data in your observations table compute the average and standard error (based on the sample standard deviation) for each value of the sum and record it in the appropriate column in your Google spreadsheet. Then, determine whether your average agrees to within +/- one standard error with the theoretical expectation for 1000 dice throws.

In the space below, answer the following two questions:

1. How many times (out of 16 possibilities) did your average agree with the theoretical average within +/- one standard error?
2. Assuming that the errors are normally distributed, how many times would you expect the average to be within +/- one standard error of the theoretical result? Hint: This was covered during today’s presentation!

|  |
| --- |
| 11/16 = 68.75 were within 1 standard error of the theoretical result. 68% is what we expect  I would argue from our small sample size, our data does not look overtly ridden with outliers. |

# Submit files to Canvas

Upload the following files to Canvas:

1. Your modified C++ program (called Lab01.cpp)
2. The shell script you wrote for this lab (called Lab01.bash) to Canvas.