CS 3360 Programming Assignment 1

Due: 11:59pm Thursday, Sep. 28, 2023

Instructor: Kecheng Yang (yangk@txstate.edu)
Doctoral Instructional Assistant: Ishrak Ratul (ishrakratul@txstate.edu)

Instructions

You can write your program in any of these languages (C, C++, Python or Java), however, it is your responsibility to ensure it compiles and runs under the **CS Department Linux server (zeus.cs.txstate.edu)** with a command line (i.e. no GUI, no IDE). Please include a README to indicate clearly how to compile and run your program. **Submissions that do not compile as described above may not receive credits**, even if it compiles and runs in other environment such as various IDEs.

Your submission should include:

- **SOURCE CODE**;
- **README** text file, which indicates how to compare and run your program, including all command lines to compile, run, and test your code on zeus.cs.txstate.edu (i.e., starting from the first command right after log in on zeus.cs.txstate.edu);
- A **REPORT** in MS Word or PDF format that briefly describes your submitted files and programs relating to the problems, report the results by your program, and answer the questions if there are any asked in the problems.

NOTE:

In all programming assignments in this course, it is NOT allowed to use any library or package function to directly generate random numbers that follow Poisson Distribution, Exponential Distribution, or any other advanced distributions. --- Writing such random number generators is a main purpose of this assignment.

The only exception that you are allowed to directly use is the ones that generate random numbers that follow a Uniform Distribution. E.g., `rand()` in C.

Problem 1 [60 pts]

Using a pseudo random number generation function (e.g., `rand()` in C or other equivalent functions in other languages) that generates uniformly distributed random numbers, generate a workload for a system that is composed of 1000 processes. You can assume that processes arrive with an expected average arrival rate of 2 processes per second that follows a Poisson Distribution and the service time (i.e., requested duration on the CPU) for each process follows an Exponential Distribution with an expected average service time of 1 second. Your outcome would be printing out a list of tuples in the format of *<process ID, arrival time, requested service time>*. You can assume that process IDs are assigned incrementally when processes arrive and that they start at 1.

Based on your actual experiment outcome, also answer the following question: what are the **actual** average arrival rate and **actual** average service time that were generated?

Problem 2 [40 pts]

A computing system is composed of two servers that are mirrors of each other (for redundancy, so if one fails, it can be restored from the other). Assume that each server has an expected MTBF of 500 hours and its continuous uptimes (the average of which is MTBF) follow an Exponential Distribution. Furthermore, assume that when a server fails, it takes exactly 10 hours to restore the data from the mirror.

(a) Write a program that generates synthetic data showing the failure and restoration times for each server over 20 years. You can assume it is always exact 24 hours per day and exact 365 days per year. **[20 pts]**

(b) Find out how long it would take until the whole computing system fails (that is when both servers happen to fail within the 10 hours restoration time; in other words, when the two servers have any restoration time durations overlapped). You would need to simulate this multiple times with different seeds for the `rand()` function (or, equivalent uniform (pseudo)random number generating function in the programming langue of your choice) and compute the average. **[20 pts]**