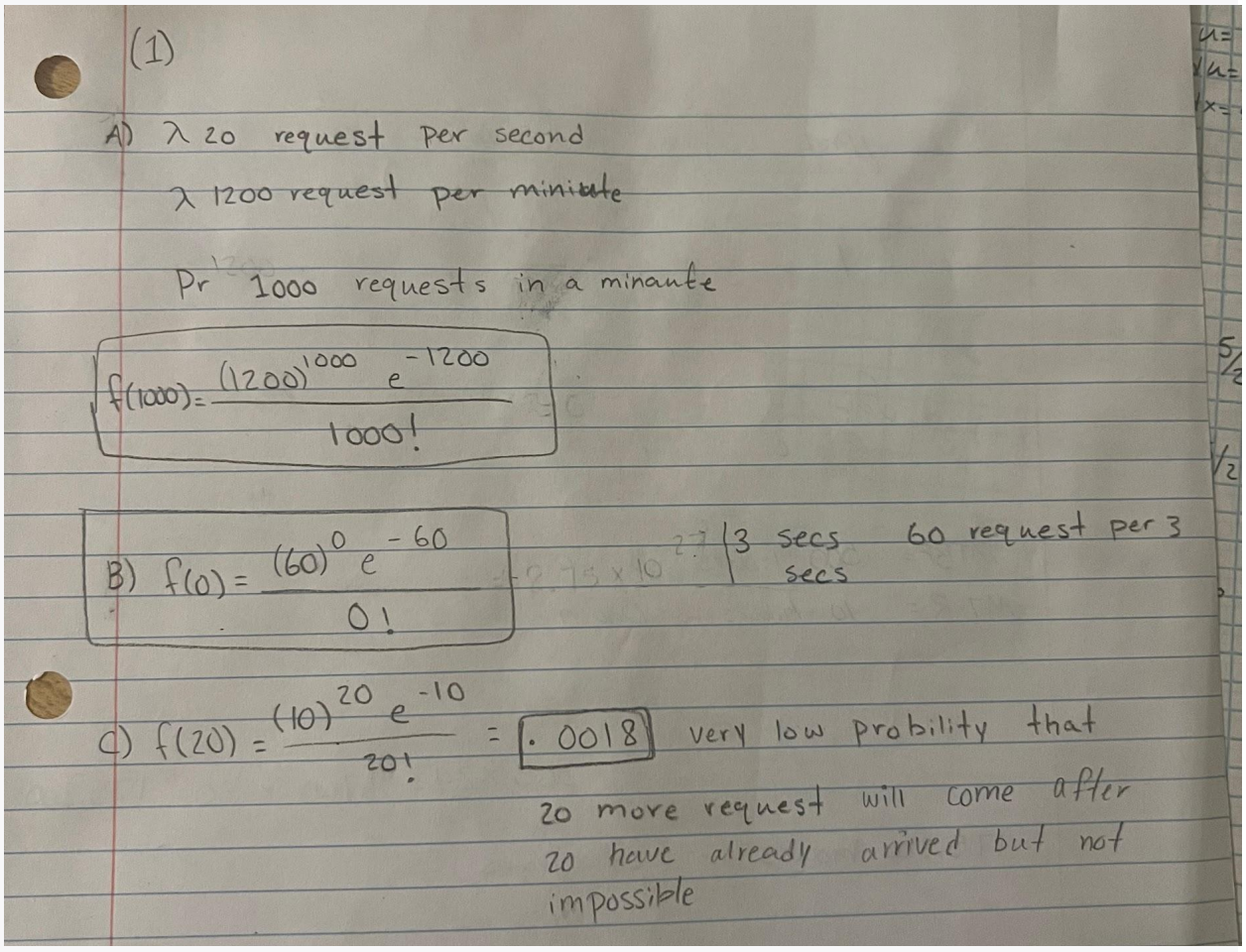
Problem 1

Consider a web-server that receives web requests at an average rate λ of 20 requests per second that follows a Poisson distribution. Answer the following questions: [2 pts each]

(a) What's the probability that it would receive 1000 requests in a minute?

(b) What's the probability that no requests will arrive for 3 seconds?

(c) Consider one second and assume that 20 requests already arrived in 0.5 second, what is the probability that 20 requests will arrive in the next 0.5 seconds?



Problem 2

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, write three functions that generate the following: [3 pts each]

(a) 100 uniformly distributed integers between 0 and 99.

(b) 100 uniformly distributed floating numbers between 0.25 and 0.5.

(c) 100 numbers in which the number 1 is produced with probability 0.5, the number 2 with probability 0.2, otherwise a floating number between 3 and 4.

See Problem2.py and Problem2.txt in the files submitted

Problem 3

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 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 average service time of 1 second. Your outcome would be <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. What is the actual average arrival rate and average service times that were generated? [30 pts]

See Problem3.py and Problem3.txt in the files submitted

Problem 4

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 MTBF of 500 hours that follows 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. [5 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). You would need to simulate this multiple times with different seeds and compute the average. [10 pts]

See Problem4.py and Problem4.txt in the files submitted