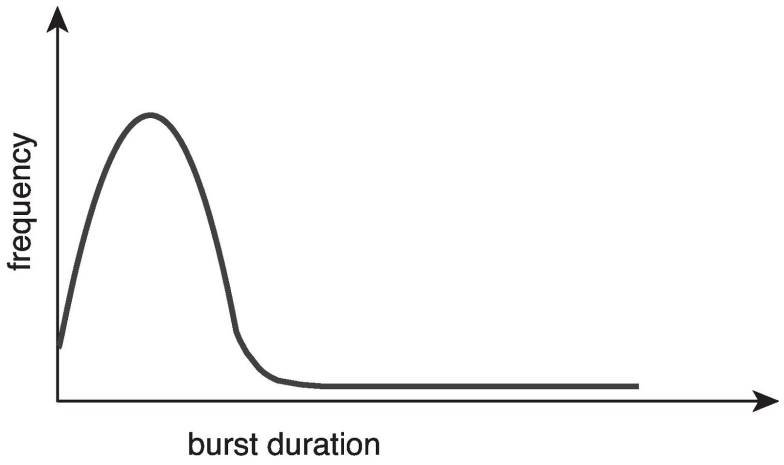
Assignment 5 (this assignment will have a 1.2 times contribution compare with regular one to your assignment grading session)

Due time: June 30, 11:59 pm

A: CPU burst time simulation (20 points)



As in the lecture, the CPU burst time will follow a long tail distribution. Please design a random number generator which can:

1. Generate N random numbers
2. The Random number follows a long tail distribution

What you should submit

1) Code used to generate those random numbers by following a **lognormal distribution.** (Please put comments in your code, without comments will get point deduction)

   API:

Parameters:

N: integer,
number of random numbers need to be generated.

mean : float

Mean value of the underlying normal distribution

sigma : float, > 0.

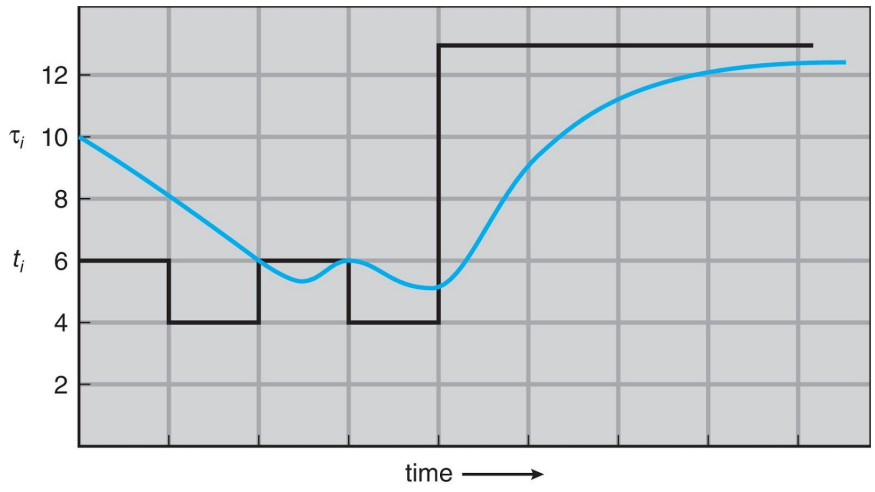
Standard deviation of the underlying normal distribution

2) Plot the histogram of the output of your random number generator by setting up your values in the following way:
   N = 1000
   Mean = 3.0
   Sigma = 1.0

Reference:
If you are using numpy:
https://docs.scipy.org/doc/numpy-1.10.1/reference/generated/numpy.random.lognormal.html#numpy.random.lognormal
If you are using C++:
http://www.cplusplus.com/reference/random/lognormal_distribution/

B :CPU burst time simulation (20 points)

Using the random number generator to generate 200 random CPU bursts.
Implement the “Prediction of the Length of the Next CPU Burst” algorithm which is “**exponential averaging**” by trying alpha =0.5 and alpha =0.1

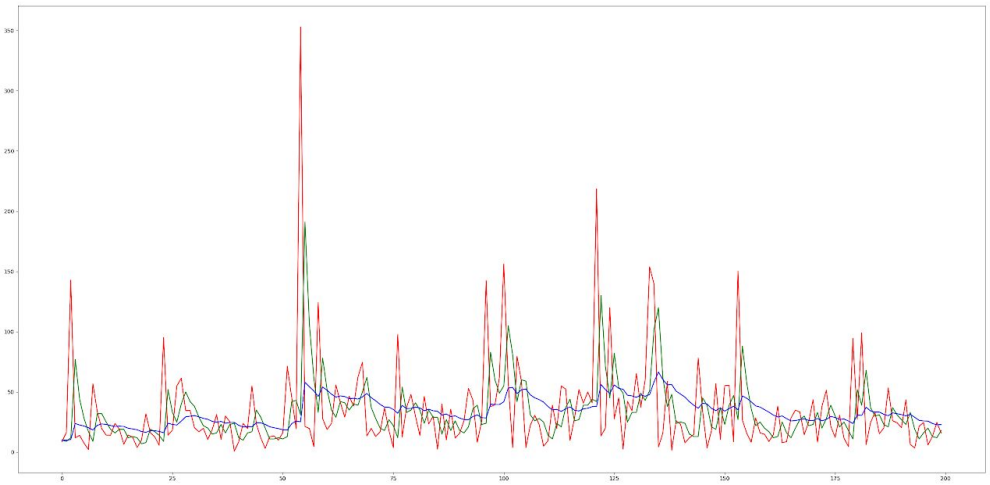


CPU burst (t_i)		6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	5	9	11	12	...

What you should submit

1) Code used to implement the "**exponential averaging**" algorithm. The visualization code is not needed.
2) Visualization of your results include: raw simulated CPU bursts. Prediction result based on alpha=0.5, prediction result based on alpha=0.1

Sample visualization output for reference:
The red one is the generated CPU bursts
The green one is prediction result when alpha=0.5
The blue one is the prediction result when alpha=0.1



C :CPU scheduling algorithms(60 points)

Collect your test results to this table:

Table 1 test results

Algorithm	Result for data size 5	Result for data size 20	Result for data size 10,000
FCFS			
SJF			

Shortest-remaining-time-first			
Round Robin			
Priority Scheduling			
Priority Scheduling with Aging			

FCFS

1) Explain this algorithm(4 points)

2) Implement this algorithm: inputs: CPU bursts, output: average waiting time (5 points)

3) Test your implementation: see how to test.

SJF

1) Explain this algorithm(4 points)

2) Implement this algorithm: inputs: CPU bursts, (To make this algorithm simple, **all processes are assumed to arrive at time 0)**. output: average waiting time(6 points)

3) Test your implementation: see how to test.

Shortest-remaining-time-first

1) Explain this algorithm(4 points)

2) Implement this algorithm: inputs: CPU bursts,arrival time for each process (we may have multiple process arriving at the same time), output: average waiting time(7 points)

3) Test your implementation see how to test.

Round Robin

1) Explain this algorithm(4 points)

2) Implement this algorithm: inputs: CPU bursts, (To make this algorithm simple, **all processes are assumed to arrive at time 0),** time quantum**,** output: average waiting time(6 points)

3) Test your implementation see how to test. (During test, set the time quantum as 6)

Priority Scheduling(If processes have the same priority, schedule the shortest remaining time one. For priority, the smaller the number, the higher the priority.)

1) Explain this algorithm(4 points)

2) Implement this algorithm: inputs: CPU bursts, (To make this algorithm simple, **all processes are assumed to arrive at time 0),** priority**,** output: average waiting time(6 points)

3) Test your implementation see how to test.

Priority Scheduling with Aging(If processes have the same priority, schedule the shortest remaining time one. For priority, the smaller the number, the higher the priority.)

4) Explain what is aging(4 points)

5) Implement this algorithm:

   For aging, you don't need to design your own aging algorithm, please use this one:

   Priority = initial_priority - waiting time.
   As priority may go to a negative value, we should set a protection in your code:
   If Priority < 0: priority = 0

inputs: CPU bursts, (To make this algorithm simple, **all processes are assumed to arrive at time 0),** priority**,** output: average waiting time(6 points)

6) Test your implementation see how to test.

How to TEST for question C

For each question, we design 3 test cases with a number of data sizes as 5, 20 and 10,000. We will have files as below:

```
arrival_time_10000.txt
arrival_time_20.txt
arrival_time_5.txt
cpu_burst_10000.txt
cpu_burst_20.txt
cpu_burst_5.txt
priority_10000.txt
priority_20.txt
priority_5.txt
```

You should read and input those test cases data to your code and collect results and put in the table 1.
For example, for FCFS algorithm, your input will be read from the file:
`cpu_burst_10000.txt`
`cpu_burst_20.txt`
`cpu_burst_5.txt`
And collect outputs for each one.

Grading rubric for question C

1. No comments in your code will get 5 points deduction in total.
2. Coding part will be graded based on your output correctness of 3 test cases (please put your results in table 1)
3. If you fail the test cases, partial credit will be given based on your implementation code.
4. If you pass all the test cases with hard coding, you will get deduction for the corresponding session.