Assignment 4: Queues, simulation of a round robin algorithm (April 10, 2020)

- Create a Queue of size hundred using linked lists. Each struct corresponds to a *job* waiting for a "processor" to process it. Thus, an essential field in each struct is the amount of processor time that is being requesting of the processor. The queue would therefore look like: J1:234; J2:123; J3:213;... J100:95, where the J* stands for the job number, and positive integer after the colon stands for the amount of processor time that this job needs.

  Generate these amount of time uniformly at random between 0 to 300 microseconds (μ). Implement the Round Robin scheduling policy that we discussed in the class to service the jobs in this queue:

  `https://en.wikipedia.org/wiki/Round-robin_scheduling`

  Let the time quantum be 50 μs. Every time a job gets serviced for a time quantum of 50 μs, it gets dequeued, and enqueued at the end, with the remaining time it needs to be serviced.

- Imagine next that a new batch of jobs gets created and appended at the end of the queue after every 200 μs. The size of the new append is half of the previous append, starting with 64. The first time around, the new append is 64 long, then 32, and so on, upto 1. Each append is generated in the same fashion as the original Queue. For example, if the original Queue is Queue1, we generate Queue2 of size 64 after 200 μs, and append it to Queue1.

  The round robin policy gets implemented on the new, larger queue–which means that a job that has been serviced for 100 μs, but still needs some more processor time, gets appended to the end of the larger queue. Implement this new "dynamic" scheme.

There is no input to these programs. When you run the first program, a static queue of jobs is created, and serviced in a round robin fashion. The second program is similar, except that after every 200 seconds, more jobs are appended at the end of the queue.

The outputs would be:
Initial Queue: J1:T1, J2:T2, ... J100:T100
Queue(1) //Queue after first 50 μs
Queue(2) //Queue after first 50 μs
⋮
Queue(i) //Queue after first 50 μs
⋮

Important: You need not simulate the processing of the job itself–for example, if a job that needs 125 μs gets the processor for the first time, directly subtract 100, reduce it to 25 μs and place it at the end of the queue.