### Data structures

**Assignment 3**: Queues Simulation

**Total points**: 100

**Due date**: March 6, 2020

Possible bonus points: use actual system clock for job arrival time, processing time, and length of time – **10 bonus points**. In which case make sure that your job is long enough to be easily reported by system time, for example: each job requires running recursive algorithm 2000 units of time. The code for such recursive algorithm doesn’t need to be your own code.

Breaking the code into modules: using classes structures or separate files for each object in the system. For example, clock has a separate file or class, CPU as structure or class in different file, job as structure or class in a separate file, statistic structure or class in separate file, queue structure or class in separate file. (**10 points**)

Using multithreading (**10 bonus points**)

Using Jason file for major input/output **10 points**

Using n CPUs instead of limited numbers of CPUS 5 points

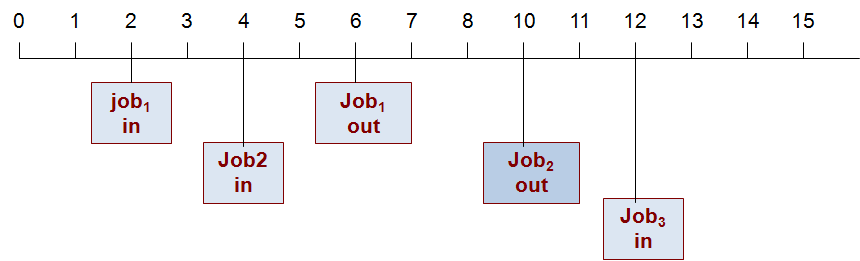
Each CPU runs at a different speed 5 bonus points

### Discrete-Event Simulation Using Queues

Create the necessary code in C++, C#, C, Java, or Python to simulate a computer system with 4 CPUs to process jobs of various lengths arriving randomly. Assume all CPUs will operate at the same speed. Use a queue to schedule the various arriving jobs according to the following guidelines.

* The clock time and the job size are represented by simple integers.
  + Jobs are of different sizes ranging from: 1-10 (generate a random number between 1 and 10 to represent job size: how many CPU time slots are needed to process the job)
  + Total clock time is between 1-600: where 1 is when you start the system and 600 is the last time to be executed.
  + Test the code with different number of jobs. For example: 100, 200, 300,400,500, 600,..,and 900. (increment by 100)
* When the computer system is first turned on (when clock time 0), all CPUs are idle and ready to start serving jobs with no waiting time.
* A new job arrives at a specific time (t1: uses integer clock: select an ascending integer to represent job arrival time). Note that job n +1 must always arrive at the same time or after jobn. (For example, Job2 must arrive after job1 or at the same time like job1 but not before it).
* All jobs will enter the same initial queue. Then each job in the queue will be served either immediately if one of the CPUs is available or it sits in the queue waiting for its turn to be served when by one of the CPUs.
* It is possible that none of the four CPUs is free to serve an arriving job. In which case there is a single queue to store the information about jobs waiting to be served by the next available CPU.
* An arriving job proceeds to the back of the queue and waits for its turn.
* The next available CPU will serve the next job waiting at the front of the queue.
* If more than one CPU is free, one of the available CPUs will be randomly selected to immediately process the next job. The CPU status will change to busy until the running job is terminated after it is completed.
* A CPU is no longer assigned to job when the job is finished (based on job size). All related information will be stored bout that job.
* Track total turnaround time for each job which is = job processing time + job waiting time in the queue.
* Example: job1 arrives at the time: 2 minutes after system starts.
  + The job requires 4 minutes.
  + Job2 arrives 4 minutes after the system starts. Will need 6 minutes for transaction.
  + Job 3 arrives 12 minutes after the system starts and needs 10 minutes.

Visual example of job execution time:



Sample input: write a function to generate the input and test it with various input sizes (100..900 , use 100 increments) to illustrate variation in the output parameters.

|  |  |  |
| --- | --- | --- |
| **Job ID** | **Ta: Arrival Time**  **As an integer** | **Tp: CPU Processing time**  **of a job** |
| 1 | 2 | 4 |
| 2 | 4 | 6 |
| 3 | 12 | 10 |
| 4 | 12 | 7 |



**Deliverables**

Zip and submit your code and all external files (sample input and output data files) to blackboard. Capture all possible scenarios and different number of jobs being executed and safe the screen capture into a Microsoft word file named with your last name + hw3.

The output consists of the following:

Calculate the following information about each job:

* The average turnaround time: The time required from when a job arrives until it completes
* Average job waiting time
* Maximum, minimum, and average job length.

Calculate the following information about the queue:

1. Maximum queue length and the time that the queue was at its maximum length
2. Total and average wait time inside the queue for all jobs.
3. Total number of jobs waited in the queue.
4. Percentage of jobs that had to wait in the queue as it relates to the total number of jobs served by the system.

Bonus points (5)

Calculate the following information about each CPU utilization:

1. Total busy time
2. Total idle time
3. Number of jobs served by the CPU.