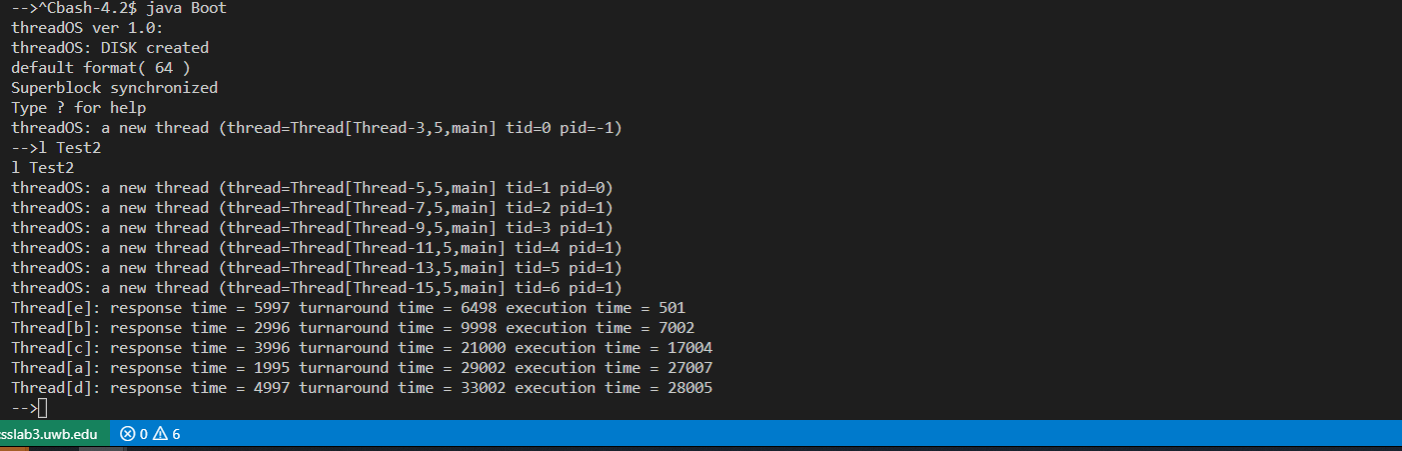
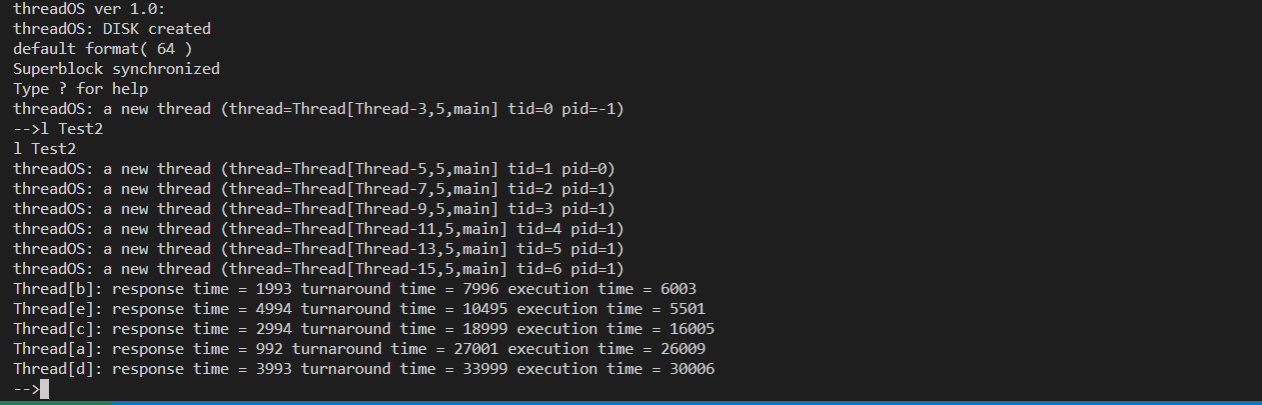
**Round Robin Vs. Multi Queue Feedback**

Round Robin Scheduling is a strict and fair scheduling algorithm. Meaning that all processes are given the same amount of quantum. They are only allowed to run for that amount of quantum before the Operating System switch them off to a new process. The benefit is that no process is being starved, and no process is going to hold up on the CPU. It is good for small tasks, the downside for Round Robin is that the average respond time is higher than its competitor. This is due to the fact of static quantum for each process.

  
*Round Robin*

Multi Queue Feedback is a dynamic scheduling algorithm. In our implementation, it utilized three distinguish queues. With each queue having different amount of quantum, process that take longer to run in first queue will move onto the second queue, and then final third queue. With MQFS, process respond time is quicker and smaller process will execute quicker and not have to wait for larger process to finish.

  
*Multi Queue Feedback*

**Analysis**

The data collected indicates that the average respond, execution, and turnaround time favors MFQS than Round Robin. The reason being is that because of the first queue’s quantum of MFQS being only 500ms, thus each thread will able to gain access to the CPU faster than the threads in Round Robin with quantum of 1000ms. Focusing on the threads’ life cycle during each scheduling algorithm, the most interesting one is Thread[e]. With Round Robin, the respond is much longer than MFQS. However, its execution time was much faster. This highlights the strength and also the weakness of RR. Although threads able to use more CPU per quantum, but if they happen to arrive late to the queue, their turnaround time is still going to be long. MFQS tries to accommodate low and medium CPU time threads by start its quantum size low and move the heavier process to different queues that have longer quantum. This is how Thread[b] was able to finish first in MFQS. MFQS respond to Thread[b] quick and was able to finish executing at the second queue. In conclusion, both scheduling algorithms has their strength and weakness. Round is fairer with each individual thread, but slower for quick CPU threads. MFQS favors the small CPU time threads first. This allow faster overall respond execution, and turnaround time, but it leaves the big CPU threads behind.

Part 2 using Queue2 FCFS.

Switching Queue2 to FCFS will allow threads that arrive at the CPU first finishes first. Because of this trait, threads that starts first will have faster respond and turnaround time. The downside of this is that if threads in front of the queue has long execution time, then it will cost congestion. Small execution time threads will have to wait until its their turn. This will increase average waiting time.

**MQFS Algorithm:**

1. Scheduler begin by initializing the scheduler object and three queues. These three queues are stored inside a vector.
2. Queue0 will get quantum of 500ms, Queue1 will get quantum of 1000ms, Queue3 will get quantum of 2000ms.
3. The scheduler will initiate run and wait for the threads.
4. When a new thread is created, it will enqueue into queue 0 if queue 0 is not empty.
5. In queue 0, it will run for 1 queue 0’s quantum. If it finishes before the quantum, it will be terminated.
6. If thread is not finish in queue 0, then it be enqueued to queue 1 and given quantum of 1000ms.
7. Thread will run for 1 quantum of queue 1. If it finishes before the quantum, it will be terminated and remove from queue 1.
8. If thread is not finish in queue 1, then it will be enqueued to queue 2 and given double of previous quantum (2000ms).
9. The threads will continue to run round-robin style until all threads are finished.