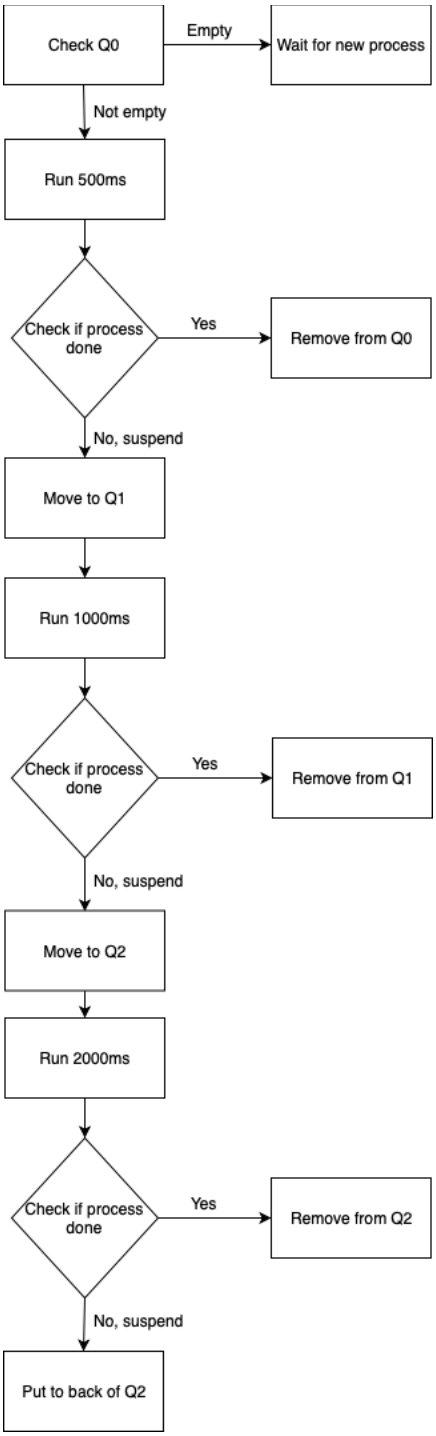
Part 1

```
gimsuyeon-ui-MacBook-Pro:ThreadOS sooyunkim$ java Boot
threadOS ver 1.0:
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,5,main] tid=0 pid=-1)
-->l Test2
l Test2
threadOS: a new thread (thread=Thread[Thread-5,5,main] tid=1 pid=0)
threadOS: a new thread (thread=Thread[Thread-7,5,main] tid=2 pid=1)
threadOS: a new thread (thread=Thread[Thread-9,5,main] tid=3 pid=1)
threadOS: a new thread (thread=Thread[Thread-11,5,main] tid=4 pid=1)
threadOS: a new thread (thread=Thread[Thread-13,5,main] tid=5 pid=1)
threadOS: a new thread (thread=Thread[Thread-15,5,main] tid=6 pid=1)
Thread[e]: response time = 6004 turnaround time = 6516 execution time = 512
Thread[b]: response time = 2999 turnaround time = 10015 execution time = 7016
Thread[c]: response time = 4001 turnaround time = 21049 execution time = 17048
Thread[a]: response time = 2000 turnaround time = 29068 execution time = 27068
Thread[d]: response time = 5001 turnaround time = 33083 execution time = 28082
```

Part 2

```
gimsuyeon-ui-MacBook-Pro:ThreadOS sooyunkim$ java Boot
threadOS ver 1.0:
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,5,main] tid=0 pid=-1)
-->l Test2
l Test2
threadOS: a new thread (thread=Thread[Thread-5,5,main] tid=1 pid=0)
threadOS: a new thread (thread=Thread[Thread-7,5,main] tid=2 pid=1)
threadOS: a new thread (thread=Thread[Thread-9,5,main] tid=3 pid=1)
threadOS: a new thread (thread=Thread[Thread-11,5,main] tid=4 pid=1)
threadOS: a new thread (thread=Thread[Thread-13,5,main] tid=5 pid=1)
threadOS: a new thread (thread=Thread[Thread-15,5,main] tid=6 pid=1)
Thread[b]: response time = 1004 turnaround time = 5529 execution time = 4525
Thread[e]: response time = 2513 turnaround time = 8037 execution time = 5524
Thread[c]: response time = 1504 turnaround time = 15105 execution time = 13601
Thread[a]: response time = 503 turnaround time = 23149 execution time = 22646
Thread[d]: response time = 2010 turnaround time = 30154 execution time = 28144
```

In my Scheduler.java from part 2, when the kernel calls the run function, the scheduler will run through a loop and check if all 3 queues are empty are not. If each queue is not empty, 3 functions: processQueue0, processQueue1, and processQueue2 will be called in this order. The program has to ensure that the highest priority queue is processed first, and the lower priority runs later.



The Round Robin (RR) algorithm in Part 1 is significantly slower in terms of response time, turnaround time, and execution time compared to the Multilevel Feedback Queue Scheduler (MFQS) algorithm in part 2. The Round Robin process takes a long time because the long response time, the turnaround time, and the execution time are so long that must wait for others to finish the queue quantum time. On the other hand, MFQS has a slightly shorter cut of quantum time for all three queues. In addition, all three queues are implemented to handle higher priority threads first, rather than keep looping them back to the back of the queue. The response time, the turnaround time, and the execution time of all threads are improved compared to part 1. The only difference between them is the way the scheduler reserves the threads.

If the last queue of the MQFS algorithm is FCFS instead of RR, the processing time for all threads in the FCFS queue is faster than RR. FCFS is used to maximize the resources imported from the system and runs until the thread is completely completed. This will result in a slight difference in which threads are completed first. When other higher priority queues come, the queue must go back and forth among them. RR focuses more on the fairness of sharing the system resources. Therefore, given that the quantum time is 2000ms, it will take longer time to finish the entire queue. Some of the threads are almost complete, but they will have to suspend and wait at the end of the queue for the next turn so that other threads can have an opportunity to use the resources. This algorithm generates a lot of context switching (suspend and resume) for all I / O and the resources in the system.

1. Copy Scheduler.java or Scheduler_part2.java to the ThreadOS folder.
2. Move to the ThreadOS directory.
3. Compile javac Scheduler.java to make java file to class file.
4. Run java Boot.
5. After '-->', type 'l Test2' to execute the thread test.