Part1) Execution output when running Test2b.java for Round Robin

```
[ratefime@sig4 prog2]$ java Boot
threadOS ver 1.0:
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,5,main] tid=0 pid=-1)
-->l Test2b
l Test2b
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[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
```

Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[e] is running
Thread[e] is running
Thread[e] is running
Thread[e] is running
Thread[e] is running
Thread[e]: response time = 6000 turnaround time = 6501 execution time = 501
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[b]: response time = 2999 turnaround time = 10001 execution time = 7002
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[a] is running
Thread[a] is running

Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[c]: response time = 3999 turnaround time = 21002 execution time = 17003
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running

Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[a]: response time = 2000 turnaround time = 29004 execution time = 27004
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d]: response time = 5000 turnaround time = 33004 execution time = 28004
Test2b finished

Part2) Execution output when running Test2b.java for MFQS

-->l Test2b
l Test2b
threadOS: a new thread (thread=Thread[Thread-5,2,main] tid=1 pid=0)
threadOS: a new thread (thread=Thread[Thread-7,2,main] tid=2 pid=1)
threadOS: a new thread (thread=Thread[Thread-9,2,main] tid=3 pid=1)
threadOS: a new thread (thread=Thread[Thread-11,2,main] tid=4 pid=1)
threadOS: a new thread (thread=Thread[Thread-13,2,main] tid=5 pid=1)

threadOS: a new thread (thread=Thread[Thread-15,2,main] tid=6 pid=1)
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[e] is running
Thread[e] is running
Thread[e] is running
Thread[e] is running
Thread[e] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b]: response time = 1000 turnaround time = 5502 execution time = 4502
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running

Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[e]: response time = 2501 turnaround time = 8002 execution time = 5501
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running

Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c]: response time = 1500 turnaround time = 16506 execution time = 15006
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a]: response time = 500 turnaround time = 24508 execution time = 24008
Thread[d] is running
Thread[d] is running
Thread[d] is running

Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d]: response time = 2001 turnaround time = 31508 execution time = 29507
Test2b finished

Part3) l test2

Round Robin:
Thread[e]: response time = 6000 turnaround time = 6500 execution time = 500
Thread[b]: response time = 2999 turnaround time = 10001 execution time = 7002
Thread[c]: response time = 3999 turnaround time = 21002 execution time = 17003
Thread[a]: response time = 2000 turnaround time = 29004 execution time = 27004
Thread[d]: response time = 5000 turnaround time = 33003 execution time = 28003

MFQS:
Thread[e]: response time = 5000 turnaround time = 5501 execution time = 501
Thread[b]: response time = 1999 turnaround time = 10004 execution time = 8005
Thread[a]: response time = 999 turnaround time = 26008 execution time = 25009
Thread[c]: response time = 2999 turnaround time = 28007 execution time = 25008
Thread[d]: response time = 4000 turnaround time = 35010 execution time = 31010

Report:

- For the second part, I was asked to change the Scheduler.java according to MFQS. For doing that, I followed the provided structures. Initially Schedular.java was implemented with one que. So, the first tasks was to implement three queues from 0 to 2 instead. After creating three queues, first I needed to change the getMyTcb

function because it was implemented only based on the one que. So, I use sychroniazed (this) to synchronize the current task and then instead of one que I search on each queues. For the constructor I used the structure for the pervious algorithm and only change the one que to have three queues instead. For the schedulerSleep() because in the run function I used it for the time quantum I had to give it the (int milliseconds) as parameter, I used timeSlice/2. For the addThread function according to the provided structure the new thread always enqueued in que0 so that is why I changed the q to que0. Other function remains the same.
for the run function:

1. First I check to see if all queues are empty, it will show the message about it.
2. According to this:MFQS scheduler first executes all threads in Queue0 whose time quantum Q0 is half of the time quantum in Part 1's round-robin. I used the exact same codes for original Schedular and implement the new one based on that. So, first I executes all thread in que0 then according to quantum Q0 is half of the time quantum in Part 1's round-robin I added schedularSleep(timeSlice/2).
3. According to this:if a thread in the Queue0 does not complete its execution for Queue0's time quantum, (Q0), then scheduler moves the corresponding TCB to Queue1. So in synchronized (qu0) I added que1.add(currentTCB) in order to moves to corresponding TCB to que1.
4. Then according to the structure, I check if que1 is not empty and que0 is empty executes the que1, I used the same code structure for que1 as well.
5. According to this: However, in order to react new threads in Queue0, your MFQS scheduler should execute a thread in queue 1 for only Q0=500ms one-half the Queue1 time quantum and then check if Queue0 has new TCBs pending. If so, it will execute all threads in Queue0 first. The interrupted thread will be resumed when all of the Queue0 threads are handled. So first I used schedularSleep(timeSlice/2) for checking the que0. Then I created the function called checkQ0 and gave it current as a parameter to check if there is new thread in que0 first re_execute que0.
6. If a thread in Queue1 does not complete its execution and was given a full Queue1's time quantum, (i.e., Q1), the scheduler then moves the TCB to Queue2 . That is why I added que2.add(currentTCB) in order to move scheduler for corresponding TCB to que2.
7. If both Queue0 and Queue1 are empty, the MFQS schedulers will execute threads in Queue2. That is why I

used while que2 is not empty and que1 and que0 is empty executes the que2.

8. However, in order to react to threads with higher priority in Queue0 and Queue1,
   your scheduler should execute a thread in Queue2 for Q0 increments and check if Queue0 and Queue1 have new TCBs. The rest of the behavior is the same as that for Queue1. So I used while loop to constantly check if there is new task in que0 and que1 and divided the timeslice by 4 in check_priority method. When it is done with executing the tasks in que0 and que1, it resumed the task for que2.
9. If a thread in Queue2 does not complete its execution for Queue2's time slice, (i.e., Q2), the scheduler puts it back to the tail of Queue2. At the end after synchronized que2 according to the struction I added que2.add.
10. Then I added 10milisecond sleep for allowing other process to execute.

- After comparing Part 1 with Part 2 through Test2.java, it's clear that MFQS often leads to shorter response times. For example, Thread[a] sees its response time cut from 2000ms in Round Robin to just 999ms in MFQS. This indicates that MFQS is quicker to start processing tasks. In my opinion, the reason behind this improvement is MFQS's ability to prioritize tasks based on their behavior and requirements, allowing urgent tasks to jump ahead and get processed sooner. Despite this, when we look at how long tasks actually take to finish (execution time), both Round Robin and MFQS show similar times. This suggests that while MFQS improves how quickly we can start working on tasks, the total time to complete these tasks doesn't change much between the two scheduling strategies. Essentially, MFQS enhances the system's responsiveness without significantly altering the speed at which tasks are completed.

- Switching Part 2 to use FCFS for Queue2 with a quantum=2000ms would impact task handling in Test2.java. The Tasks would be processed by arrival order, potentially delaying later tasks, even short ones. This could lead to longer response times in Queue2, as tasks wait their turn without the time-slice. While FCFS simplifies processing and could benefit the earliest tasks or during low load, it might reduce overall system responsiveness to new tasks in busier times.