Ryan English

MFCW

```
-->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 = 1999 turnaround time = 9001 execution time = 7002
Thread[e]: response time = 5000 turnaround time = 11501 execution time = 6501
Thread[a]: response time = 998 turnaround time = 30507 execution time = 29509
Thread[c]: response time = 2999 turnaround time = 39508 execution time = 36509
Thread[d]: response time = 3999 turnaround time = 40508 execution time = 36509
```

RR

```
-->l Test2
l Test2
threadOS: a new thread (thread=Thread[Thread-17,5,main] tid=7 pid=0)
threadOS: a new thread (thread=Thread[Thread-19,5,main] tid=8 pid=7)
threadOS: a new thread (thread=Thread[Thread-21,5,main] tid=9 pid=7)
threadOS: a new thread (thread=Thread[Thread-23,5,main] tid=10 pid=7)
threadOS: a new thread (thread=Thread[Thread-25,5,main] tid=11 pid=7)
threadOS: a new thread (thread=Thread[Thread-27,5,main] tid=12 pid=7)
Thread[e]: response time = 6001 turnaround time = 6501 execution time = 500
Thread[b]: response time = 3000 turnaround time = 10001 execution time = 7001
Thread[c]: response time = 4000 turnaround time = 21003 execution time = 17003
Thread[a]: response time = 1999 turnaround time = 29004 execution time = 27005
Thread[d]: response time = 5001 turnaround time = 33004 execution time = 28003
```

Implementation of MFQS:

getMyTcb() was edited from only checking the only queue to checking queue0 for the designated thread, and then checking queue1, and then checking queue2. If the thread was not found in any of these queues, it returned null.

When the Scheduler was initialized, it now initializes timeSlice to 500ms, and also initializes queue0, queue1, and queue2.

addThread(Thread t) was changed so that if a new thread is added it is added to queue0 and not queue

run() was modified to first check if queue0 was empty. If it is not empty, it will set the TCB to the first element. It then checks if the thread's dead, and if so removes it. If not, it checks if it's alive. If it's alive, this means that the program has looped already and the thread has outlasted the designated quantum

assigned for queue0 of 500ms. Therefore, it is removed from queue0 and added to queue1. If the thread was never started it is started.

After checking queue0, it will check queue1. It assigns queue1 to the current thread in execution. If there is no thread (i.e. getMyTcb() returns null), then it assigns currentTCB to the first item in queue1. After checking if TCB is terminated, it will check if the current thread is alive and if the value repeated is less than 2. The value repeated increments every time the current's execution is repeated, but will only go up to 2 for queue1 since queue1's time quantum is 1000ms, which is 500*2. Therefore, if the thread has executed less than 2, that means that queue1's time quantum has not been reached. However, once the repeated==2, it is reset to 0 and the current thread is paused, removed and moved to queue2.

After checking queue0 and queue1, it will check queue2. Similarly to queue1, it will check if there is a thread in execution. If not, it will assign TCB to the first item in queue2. The checks for queue2 are the same as queue1 except that it checks for repeated to be 4, since queue2's time quantum is 2000ms, or 4*500 ms. If 2000ms or repeated = 4 is reached, the thread is moved to the back of the queue.

The MFQS was overall faster than the RR scheduler. The reason being is that while RR brute forces its way through each thread execution, MFQS takes out the smallest threads first in smaller bites first, while distributing the rest of the threads to be taken out in increasingly larger bites until it is left with the leftovers of several large threads, in which case it brute forces the threads in large chunks.

If the First Come First Serve method was implemented instead of round robin, if a program utilizes multiple threads the threads in the back would never see execution until the threads in the front are finished. This means that while the scheduler is brute forcing its way through the initial threads the threads in the back would most likely have to wait several seconds to start execution. Compared to the Round Robin style, every thread would have started execution in the beginning, allowing for the CPU to take on more threads at time.

MFQS Test2b output:

```
-->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[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[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] 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 = 1999 turnaround time = 8503 execution time = 6504
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 = 5000 turnaround time = 11001 execution time = 6001
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] 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[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]: response time = 999 turnaround time = 30008 execution time = 29009
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[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c]: response time = 3000 turnaround time = 39009 execution time = 36009
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d]: response time = 4000 turnaround time = 40009 execution time = 36009
Test2b finished
```

RR Test2b Output:

```
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[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[a] is running
Thread[b]: response time = 2999 turnaround time = 10000 execution time = 7001
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] 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[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[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 = 1999 turnaround time = 29003 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]: response time = 4999 turnaround time = 33003 execution time = 28004
Test2b finished
```