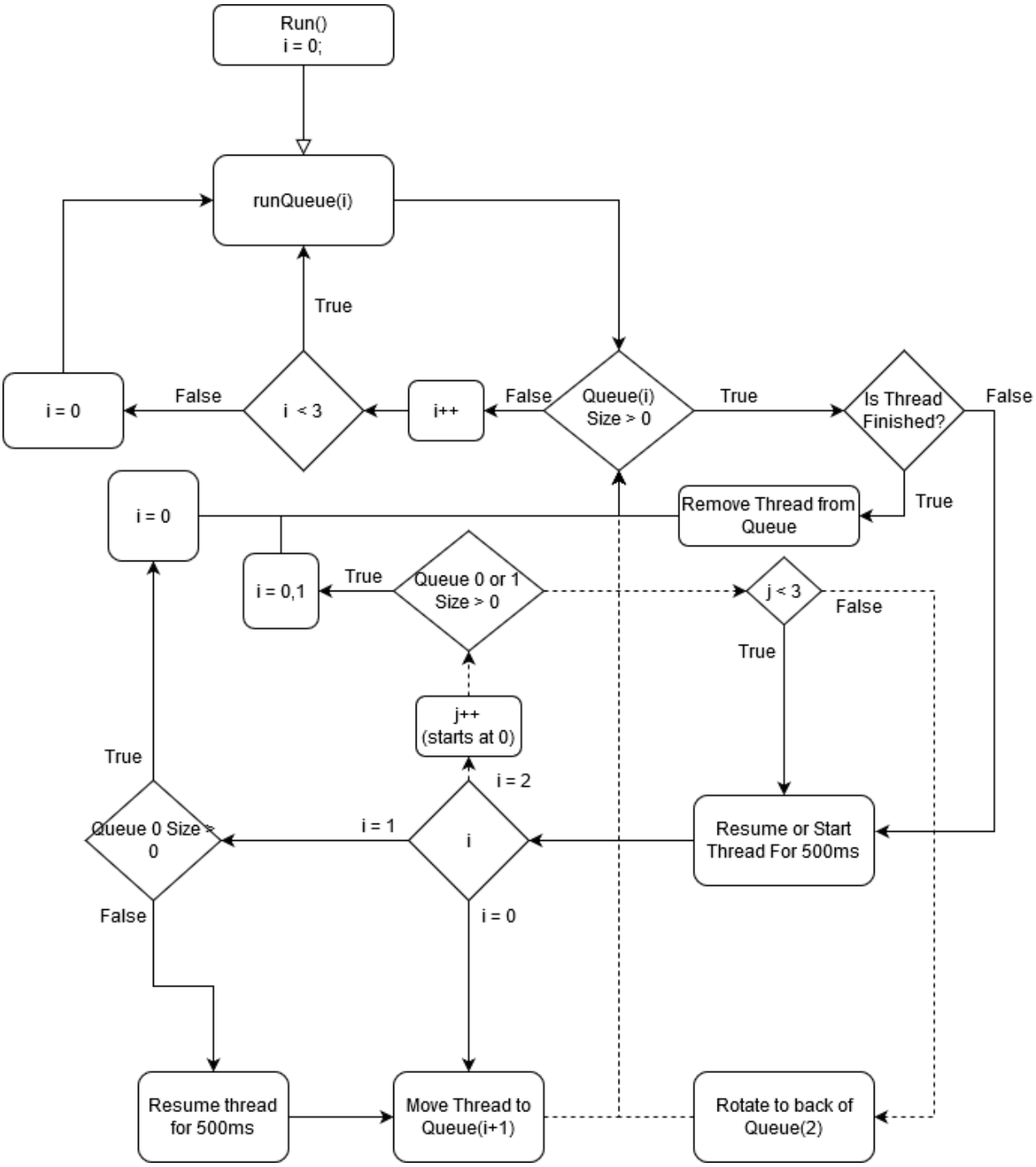
Colton Sellers
CSS 430
Program 2: Scheduling

Multilevel Feedback Queue Scheduler Algorithm



The MFQS Algorithm is a complex algorithm that prioritizes new threads in the queue zero. The figure above shows the steps of its scheduling between three queues, each a lower priority than the previous.

Comparison of Algorithms

TEST 2 OUTPUT FOR ROUND ROBIN

```
shell[2]% Test2
Test2
threadOS: a new thread (thread=Thread[Thread-19,5,main] tid=8 pid=1)
threadOS: a new thread (thread=Thread[Thread-21,5,main] tid=9 pid=8)
threadOS: a new thread (thread=Thread[Thread-23,5,main] tid=10 pid=8)
threadOS: a new thread (thread=Thread[Thread-25,5,main] tid=11 pid=8)
threadOS: a new thread (thread=Thread[Thread-27,5,main] tid=12 pid=8)
threadOS: a new thread (thread=Thread[Thread-29,5,main] tid=13 pid=8)
Thread[e]: response time = 6996 turnaround time = 7498 execution time = 502
Thread[b]: response time = 3994 turnaround time = 11998 execution time = 8004
Thread[c]: response time = 4994 turnaround time = 25004 execution time = 20010
Thread[a]: response time = 2994 turnaround time = 35010 execution time = 32016
Thread[d]: response time = 5996 turnaround time = 40011 execution time = 34015
```

TEST 2 OUTPUT FOR MULTI LEVEL FEEDBACK QUEUE SCHEDULER

```
shell[1]% Test2
Test2
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=2)
threadOS: a new thread (thread=Thread[Thread-11,5,main] tid=4 pid=2)
threadOS: a new thread (thread=Thread[Thread-13,5,main] tid=5 pid=2)
threadOS: a new thread (thread=Thread[Thread-15,5,main] tid=6 pid=2)
threadOS: a new thread (thread=Thread[Thread-17,5,main] tid=7 pid=2)
Thread[b]: response time = 994 turnaround time = 5501 execution time = 4507
Thread[e]: response time = 2489 turnaround time = 7996 execution time = 5507
Thread[c]: response time = 1489 turnaround time = 17512 execution time = 16023
Thread[a]: response time = 500 turnaround time = 27534 execution time = 27034
Thread[d]: response time = 1990 turnaround time = 36529 execution time = 34539
```

From the results its clear that MFQS tends to preform better on average than a round robin implementation. This is due to the fact that MFQS is continuously looking to its newer threads to ensure response time and turnaround time are much lower on average. This approach also quickly addresses new threads that have may have a small execution time thus improving overall performance by getting these tasks out of the way.

Even though in some instances the execution time is higher than the round robin implementation it does prove to be a more optimized algorithm overall.

FCFS Queue for Part 2 Queue 2

If I were to implement a FCFS for Queue 2 in part 2 of this assignment I would expect the overall execution times to be slightly improved for the threads that made it to that queue. That being said the turnaround times would increase greatly at that point as it would only address the earliest in the queue.