In total, we implemented eight algorithms for process scheduling: first-come first-serve (FCFS), shortest job first (SJF), shortest remaining time (SRT), round robin (RR), and four variants of highest-priority first (HPF)—preemptive with and without aging, and non-preemptive with and without aging. These algorithms all have their strengths and weaknesses with regard to the different metrics we use to measure the performance of scheduling algorithms. We calculated four primary metrics: turnaround time, waiting time, response time, and throughput.

Turnaround Time: SJF (26.65444 quanta)
Waiting Time: SJF (21.76286 quanta)
Response Time: RR (7.74492 quanta)
Throughput: SJF (0.21562 processes/quanta)

Turnaround Time
The shortest job first algorithm has the shortest average turnaround time, closely followed by shortest remaining time. This makes sense, as the shortest job first algorithm schedules longest jobs later, so short jobs do not have to wait on long jobs to be completed, improving the overall turnaround time.

Waiting Time
Once again, shortest job first has the lowest average waiting time. This is the expected result—the overall waiting time is minimized since no one has to wait on the longest jobs to finish, since the longest jobs are scheduled last. Additionally, being a non-preemptive algorithm, the only time a process waits is after it has arrived but before it has started. Thus, the waiting time is equal to the response time.

Response Time
The round robin algorithm does the best job by far of optimizing for response time. By preempting every 5 quanta, it guarantees a worst-case response time of (# jobs) × 5, while the average response time is much lower. Of the other algorithms, the shortest job first algorithm provides the best average response time, closely followed by the shortest remaining time algorithm. This is for the same reason that the two algorithms optimize well for turnaround time: they avoid scheduling shorter jobs after longer jobs.

Throughput
Shortest job first also had the highest throughput. This was a given since it had the lowest average turnaround time, so it obviously had to be going through more processes per quanta than any other algorithm.

The Highest Priority First Algorithms
All four of the highest priority first algorithms fall short of the others in the above benchmarks as averaged across all processes. However, they are the only ones to take the priority of a process into account. While only considering processes running at priority 1 (the maximum), preemptive HPF performs the best in all metrics except throughput. In an HPF-scheduled

system, high priority processes will get significantly better performance, but lower priority processes will be worse off.

The versions of HPF that employ aging appear to perform poorly even for the processes at priority 1. This is due to an error in the way our data was recorded. Averages for each metric were taken with respect to which priority queue a process was in when it finished executing, rather than when it began executing. As a result, low-priority processes that gradually aged to the higher priority queues skewed the numbers. The expected behavior of the HPF algorithms with aging was for better performance for the lower priorities at the cost of slightly worse performance for the higher priorities.

Other Points of Note

Shortest job first and shortest remaining time were very close in every metric. This makes sense, as they are essentially the same algorithm, but with and without preemption. Shortest remaining time should, in principle, have better waiting times than shortest job first; however, due to the restrictions on the simulation (only 100 quanta, limits on expected running time of jobs), this did not bear out in practice. In general, shortest job first and shortest remaining time were quite a bit better than all the other algorithms for every metric except for response time.