Intro:

In this part of the assignment we made a simulator that dealt with processes using 3 different algorithms. The first algorithm was external priority, round robin scheduling and the third one was a mix of external priority and the round robin algorithm. In any of these algorithms, the process first has to be put into a large enough chunk of memory other wise it becomes in a waiting state. 21 simulations were done. There are 7 input files and each of the three scheduling algorithms were tested on the 7 input files.

Scheduling Algorithms Explanation:

External Priority:

This was a non preemptive algorithm which means that the process pretty much always completed unless theres an io event that's scheduled. The lower process ids were given higher priority but they can’t really interrupt and stop the current process that was running. This algorithm worked pretty well when all the processes were around the same length. The only problem with this algorithm is that when a long process has a higher priority than other shorter processes, the long process actually hogs all the CPU time and could starve the other processes.

Round Robin:

This algorithm pretty much solves the problem with the external priority algorithm. Rather than allowing just one long process to take up all the CPU time this algorithm actually switches out the processes after a certain amount of time quantum passes (which in our code was 100 ms). This means that a process gets 100 ms to complete and if it doesn’t end another process will get CPU time and switch places with the running process. This way, a long process with high priority can’t really starve other processes because of the constant switching. Even though these external priority and round robin algorithms are different, they actually gave very similar outputs for most of the inputs. The reason for this is the time quantum being so big compared to our test processes’ length. Most of our test processes are not anywhere near 100 ms long. This means that the processes could actually go all the way to completion. This makes the behavior similar to the external priority algorithm but only for the cases where the process lengths are small compared to the time quantum.

External Priority and Round Robin:

In this algorithm preemption happens when a higher priority process comes. There is still the round robin aspect with processes with similar priorities but once a higher priority process appears, the running process gets put in the ready queue. This is actually why the output for this algorithm was more unique than the other two. This algorithm actually allows a process to interrupt another process no matter the running time of the current process.

Simulation Results:

Algorithm	Average Throughput	Average Turn Around Time(across all processes)	Average Waiting Time(across all processes)	Average Response Time
External Priority	0.055 processes/ms	28.05 ms	2.11 ms	8 ms
Round Robin	0.055 processes/ms	28.05 ms	2.11 ms	8 ms
External Priority & Round Robin Mix	0.055 processes/ms	22.6 ms	4.2	6.65

Both the external priority and round robin algorithms gave us the exact same results. This is because the round robin algorithm had a quantum time(100 ms) which was larger than any of the used process times. As mentioned before since the process times are shorter, the RR algorithm does not need to switch any process out and it behaves the same as the external priority one. On the other hand the EP and RR mix algorithm behaved slightly differently. The throughput didn’t change since the execution times and the amount of processes that completed were the exact same across all inputs. The average turn around time on the other hand went down. This is because higher priority processes were able to finish earlier under the algorithm’s behavior. This however also came with a higher waiting time since processes were being switched around in the ready queue based on priority. The response time for the EPRR algorithm was also less because of the more frequent switching caused by both the priority based and round robin based behavior. If the time quantum was 3 ms for our RR algorithm, the average response time for that would be lower than it is shown to be in the table since there would actually be alot of switching going on.