George Flouris Christos Stylianou

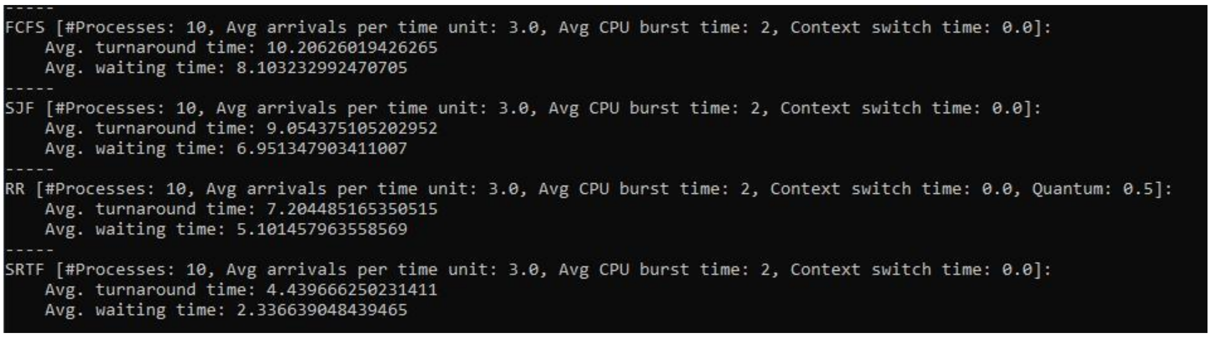
NOSE2

Report on Assessed Exercise 2: Scheduling

The following report shows detailed observations obtained from comparing the sorting algorithms implemented in Python. The four algorithms are **First Come First Server**, **Shortest Job First**, **Round Robin** and **Shortest Remaining Time First**.

By observing the given sample outputs or running the completed and correctly working code, with random seeds, we observe that in general SJF (Shortest Job First) and SRTF (Shortest Remaining Time First) algorithms usually have lower average turnaround time and average waiting time than the other 2 algorithms with the SRTF algorithm almost always being the most efficient of all. Additionally, RR (Round Robin) is more efficient than FCFS (First Come First Serve) most of the times. These observations are thus the expected outcome for any seed.

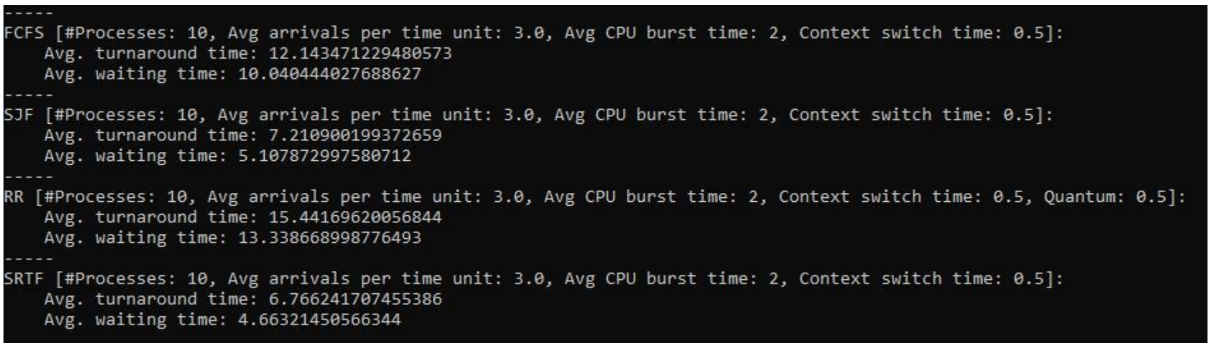
Seed 1 - 1797410758, (context switch time: 0.0):



```
-----
FCFS [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.0]:
    Avg. turnaround time: 10.20626019426265
    Avg. waiting time: 8.103232992470705
-----
SJF [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.0]:
    Avg. turnaround time: 9.054375105202952
    Avg. waiting time: 6.951347903411007
-----
RR [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.0, Quantum: 0.5]:
    Avg. turnaround time: 7.204485165350515
    Avg. waiting time: 5.101457963558569
-----
SRTF [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.0]:
    Avg. turnaround time: 4.439666250231411
    Avg. waiting time: 2.336639048439465
```

As predicted, the SRTF algorithm is the most efficient of the 4 algorithms however the results of this seed deviate from the expected as SJF has quite higher average turnaround and wait time than RR. The SJF (Shortest Job First) algorithm executes processes in a queue based on their service time. When using this seed, a process with rather large service time, arrives 3rd at the system but due to it having the largest service time out of all processes it is executed last. Since the process must wait until every other process is terminated, and it arrives rather early in the queue this process alone significantly increases the average turnaround time and average waiting time for this algorithm making it less efficient than RR but not enough to be less efficient than FCFS.

Seed 1 - 1797410758, (context switch time: 0.5):

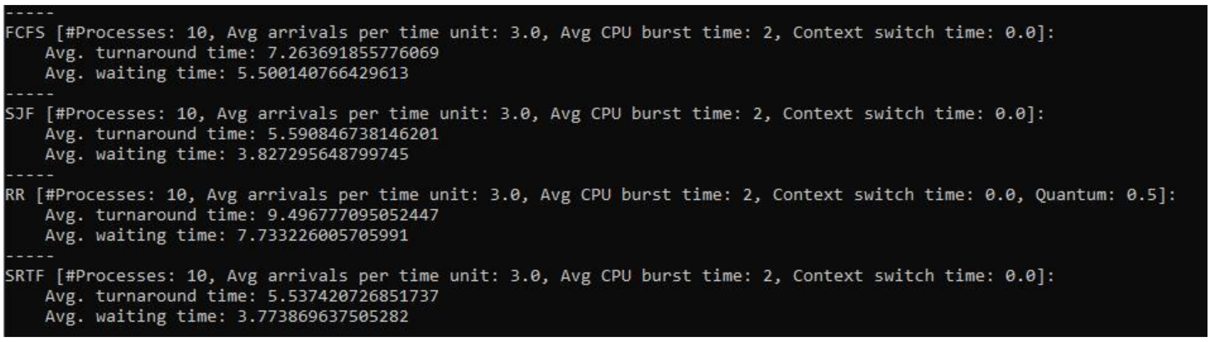


```
-----
FCFS [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.5]:
    Avg. turnaround time: 12.143471229480573
    Avg. waiting time: 10.040444027688627
-----
SJF [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.5]:
    Avg. turnaround time: 7.210900199372659
    Avg. waiting time: 5.107872997580712
-----
RR [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.5, Quantum: 0.5]:
    Avg. turnaround time: 15.44169620056844
    Avg. waiting time: 13.338668998776493
-----
SRTF [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.5]:
    Avg. turnaround time: 6.766241707455386
    Avg. waiting time: 4.66321450566344
```

Factoring in context switch time, with value 0.5, the results change significantly. We now observe that SJF is the second most efficient algorithm followed by FCFS and RR now has the largest average turnaround and waiting times. The RR (Round Robin) algorithm executes processes in the order in which they are added to the queue however it does so only for a specified amount of time (referred to as quantum) and then moves on to the next process, regardless of whether the previous process has been fully executed or not. RR is the least efficient here as with some processes requiring going through several times, due to their large service time compared to the rather small value of the quantum, the context switch time is repeatedly added to the total time.

SJF is the only algorithm that shows a rather unexpected increase in efficiency. This is due to the fact that in the added time after every context switch, processes are still arriving in the queue and thus a process with the lowest service time might now be in the queue and thus be executed next, while this process would have not been in the queue had the context switch time been 0. This effectively changes the order in which processes are executed and can thus make SJF more efficient.

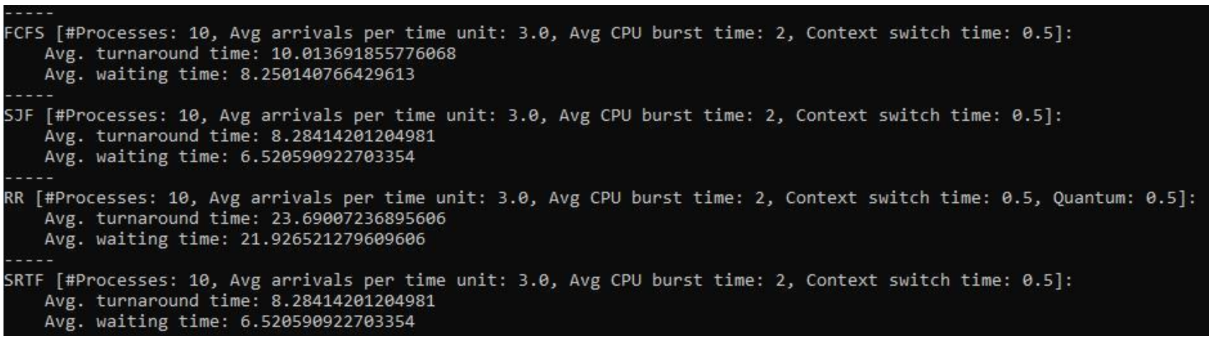
Seed 2 **–** 2688744162 (context switch time: 0.0):



```
-----
FCFS [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.0]:
    Avg. turnaround time: 7.263691855776069
    Avg. waiting time: 5.500140766429613
-----
SJF [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.0]:
    Avg. turnaround time: 5.590846738146201
    Avg. waiting time: 3.827295648799745
-----
RR [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.0, Quantum: 0.5]:
    Avg. turnaround time: 9.496777095052447
    Avg. waiting time: 7.733226005705991
-----
SRTF [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.0]:
    Avg. turnaround time: 5.537420726851737
    Avg. waiting time: 3.773869637505282
```

Once again, the SRTF algorithm is the most efficient with SJF slightly behind, however we observe that average turnaround and waiting time of RR is significantly larger than all other algorithms. The reason why RR is the least efficient with this seed is because every single process arriving at the system has service time larger than the given quantum. Thus, for all 10 processes in the queue, execution will halt at least once before each process is terminated and then go back to it again. Thus, both average turnaround time and average waiting time will increase making RR the least efficient of the 4 algorithms.

Seed 2 – 2688744162 (context switch time: 0.5):

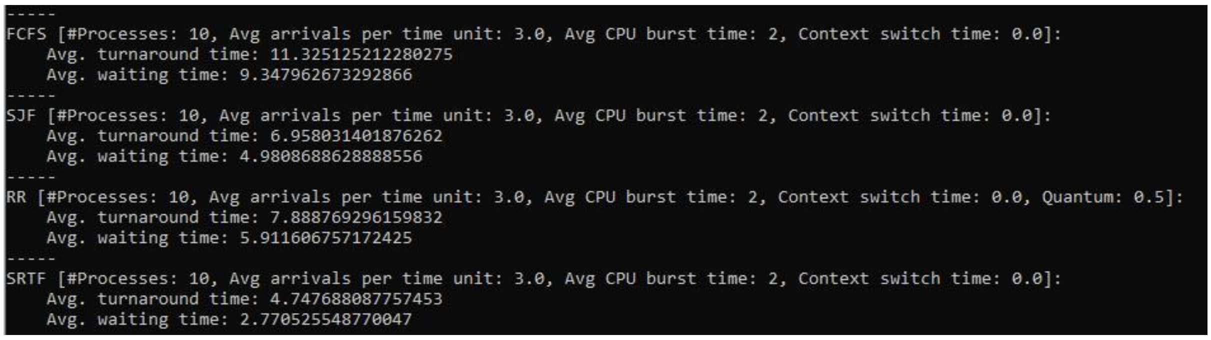


```
-----
FCFS [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.5]:
    Avg. turnaround time: 10.013691855776068
    Avg. waiting time: 8.250140766429613
-----
SJF [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.5]:
    Avg. turnaround time: 8.28414201204981
    Avg. waiting time: 6.520590922703354
-----
RR [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.5, Quantum: 0.5]:
    Avg. turnaround time: 23.69007236895606
    Avg. waiting time: 21.926521279609606
-----
SRTF [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.5]:
    Avg. turnaround time: 8.28414201204981
    Avg. waiting time: 6.520590922703354
```

As expected, with context switch time introduced, the RR algorithm shows a very large increase which is, as explained above, since execution will halt at least once for every single process before each process is terminated and then go back to it again later.

Another interesting observation would be that SJF and SRTF now have the exact same average turnaround and waiting times. The SRTF algorithm is basically the pre-emptive version of the SJF algorithm. As with the first seed, introducing context switch to the system changes the order of execution for SJF. In this case the order of termination of the processes ends up the same between the two algorithms (starting from the first process to terminate the order is 0,2,3,1,9,5,7,6,8,4 for both) and thus their average turnaround and waiting times are the same.

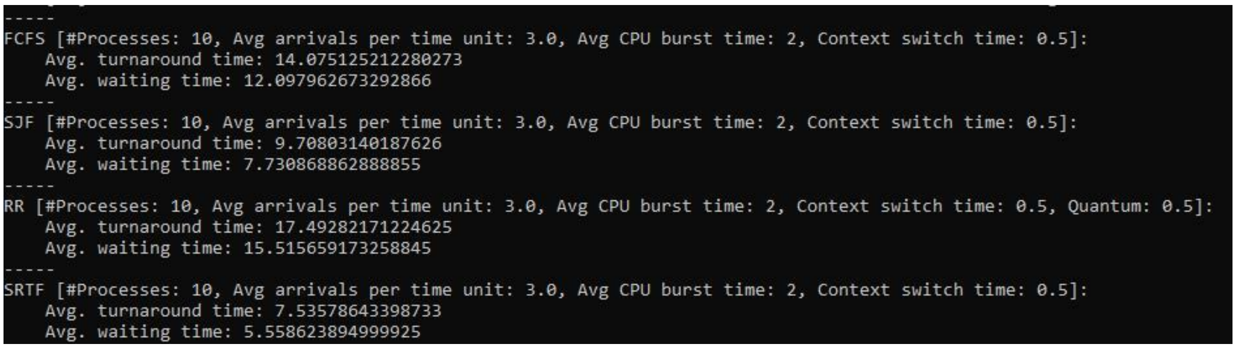
Seed 3 – 3399474557 (context switch time: 0.0):



```
-----
FCFS [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.0]:
    Avg. turnaround time: 11.325125212280275
    Avg. waiting time: 9.347962673292866
-----
SJF [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.0]:
    Avg. turnaround time: 6.958031401876262
    Avg. waiting time: 4.9808688628888556
-----
RR [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.0, Quantum: 0.5]:
    Avg. turnaround time: 7.888769296159832
    Avg. waiting time: 5.911606757172425
-----
SRTF [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.0]:
    Avg. turnaround time: 4.747688087757453
    Avg. waiting time: 2.770525548770047
```

Lastly, SRTF is the most efficient for seed 3 as well, with SJF and RR following as expected. While the order of efficiency of the 4 algorithms is as expected it is worth looking further into why the FCFS has such large average turnaround and waiting time compared to the rest. FCFS (First Come First Server) executes processes in the order in which they arrived at the system. Using this seed provides the system with a couple of processes that have rather large Service time. Following these large processes, the following arrive in quick succession and thus until the large processes are executed all following processes are waiting. This has an increase on both the average turnaround time and waiting time for the FCFS algorithm and thus making it by far the least efficient of the 4.

Seed 3 – 3399474557 (context switch time: 0.5):



```
-----
FCFS [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.5]:
    Avg. turnaround time: 14.075125212280273
    Avg. waiting time: 12.097962673292866
-----
SJF [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.5]:
    Avg. turnaround time: 9.70803140187626
    Avg. waiting time: 7.730868862888855
-----
RR [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.5, Quantum: 0.5]:
    Avg. turnaround time: 17.49282171224625
    Avg. waiting time: 15.515659173258845
-----
SRTF [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.5]:
    Avg. turnaround time: 7.53578643398733
    Avg. waiting time: 5.558623894999925
```

Context switch time of 0.5 results in a shift in the order of efficiency of the 4 algorithms. Specifically, a large increase in turnaround and waiting time is brought about in the RR algorithm making it the least efficient. This once again is due to some processes having several times larger service time than the value of the quantum. The algorithm will thus switch between these processes quite a few times until every single one is terminated with the context switch time being added every time and increasing both average turnaround time and waiting time severely.