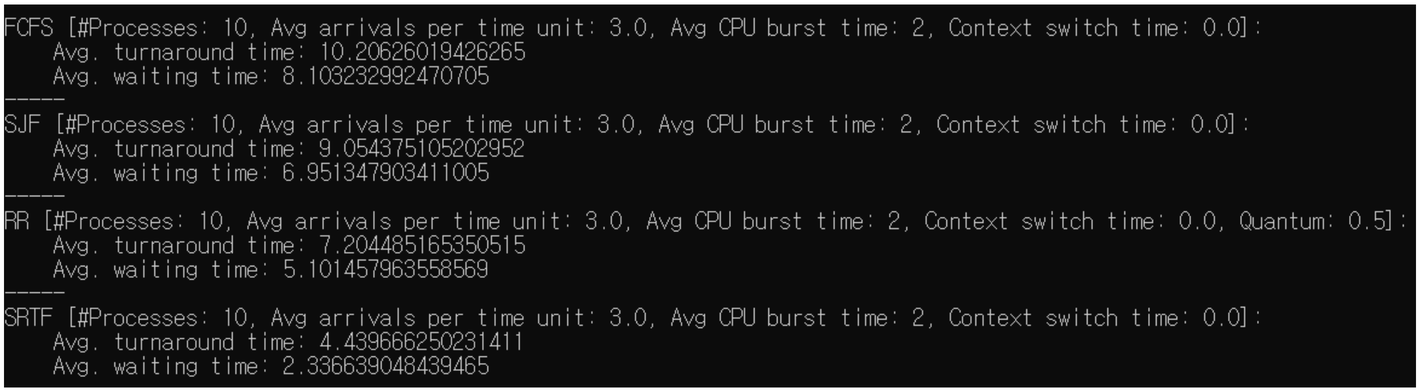
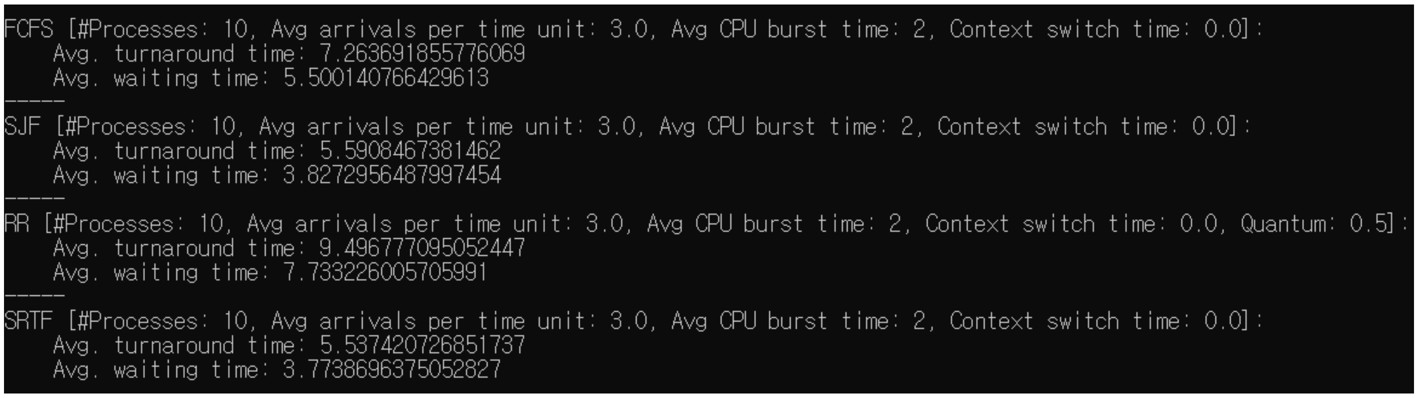
Seed used: 1797410758



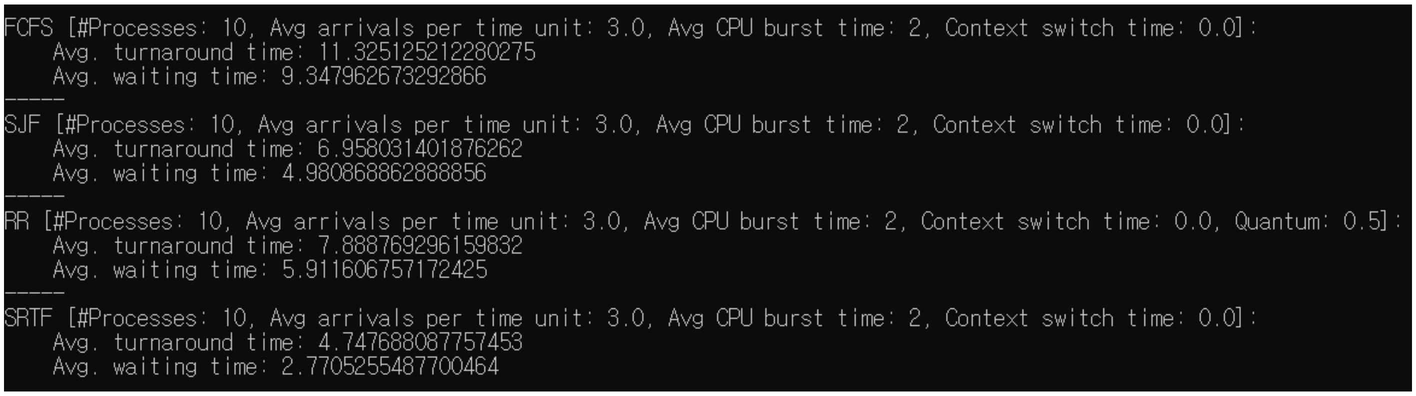
```
FCFS [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.0]:
    Avg. turnaround time: 10.20626019426265
    Avg. waiting time: 8.103232992470705
-----
SJF [#Processes: 10, Avg arrivals per time unit: 3.0, Avg CPU burst time: 2, Context switch time: 0.0]:
    Avg. turnaround time: 9.054375105202952
    Avg. waiting time: 6.951347903411005
-----
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
```

Seed used: 2688744162



```
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: 5.5908467381462
    Avg. waiting time: 3.8272956487997454
-----
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: 3.7738696375052827
```

Seed used: 3399474557



```
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: 4.980868862888856
-----
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: 2.7705255487700464
```

The relative performance of the four scheduling algorithms

The relative performance of the four scheduling algorithms were as follows: the average waiting time and the average turnaround time for the algorithms were in the order of SRTF < RR < SJF < FCFS for the first seed(1797410758), SRTF < SJF < FCFS < RR for the second seed(2688744162), and SRTF < SJF < RR < FCFS for the third seed(3399474557). SRTF algorithm had the smallest average waiting times and turnaround times for all three seeds, and the average waiting time and turnaround time of SJF was less than that of FCFS for all three seeds; but the order of the average waiting time and turnaround time of the RR algorithm was different for each seed - the average waiting and turnaround times of RR was less than both

SJF and FCFS, greater than both SJF and FCFS, and in between SJF and FCFS for the first, second and third seed respectively.
What we expected
SJF (shortest job first) and SRTF (shortest remaining time first), which is a preemptive version of SJF, both have provable minimum average waiting time. On the other hand, for both FCFS(first-come, first-served) and RR (round-robin) scheduling, the average waiting time can be quite high. Also, the waiting time of preemptive scheduling is less than the waiting time of non-preemptive scheduling.

Therefore, we expect the average waiting time of SJF and SRTF to be less than the average waiting time of FCFS and RR; the waiting time of SRTF, which is preemptive, to be less than the waiting time of SJF, which is non-preemptive; and the waiting time of RR, which is preemptive, to be less than the waiting time of FCFS, which is non-preemptive; so that the waiting times of the four scheduling algorithms are in the order of: SRTF < SJF < RR < FCFS.
The order of the average turnaround times and the order of the average waiting times of the four scheduling algorithms would be the same for each seed, as the turnaround time is simply the waiting time added to the CPU burst time (or the service time) and since the four algorithms in each seed all share the same queue of processes, and each process would have the same CPU burst time regardless of the algorithm used, so the order for the turnaround times would be the same as that of waiting times for the four algorithms.

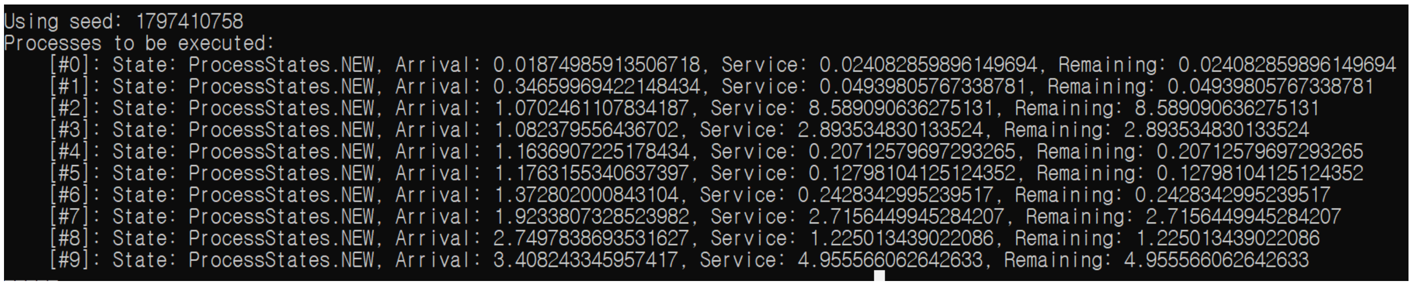
Deviation of results:
The **third seed gave the results for the average waiting times (and average turnaround times) as expected**, as the order of the average waiting times (and average turnaround times) of the four scheduling algorithms for the third seed was: SRTF < SJF < RR < FCFS. However, the first seed and the second seed did not give the average waiting times as expected.

1. **first seed - deviation and reason of deviation**

For the first seed, the average waiting times (and average turnaround times) for different scheduling algorithms are ordered as follows: SRTF < RR < SJF < FCFS. The fact that SRTF has the lowest average waiting time and the FCFS has the largest average waiting time is as expected, however the average waiting time of RR < SJF is not as expected as we have expected them to be in the opposite order.

The reason for this may be that the total service time of the processes for the first seed, second seed and third seed is 21.030272017919460074, 17.6355108934645539 and 19.771625389874062748 respectively, which means the first seed has the largest total service time for the processes. As SJF and FCFS are non-preemptive and therefore the processes cannot be interrupted in the middle, this can lead to a problem of starvation as the processes later in the queue has to wait for the whole service time(s) of the process(es) earlier in the queue to finish (i.e. wait until the process(es) earlier in the queue terminates) before itself gets executed.
This is more so because as seen below, the gap between the maximum and minimum service time of the processes is quite large for this seed, and two of the processes with the largest

service times are earlier in the queue (process 2 and 3), which means that for FCFS, where the processes in queue are ordered by the time of arrival like below in the picture and not by the CPU burst time (as for SJF), the processes later in the queue would have to wait a long time until those two processes terminate. (On the other hand, the gap between maximum and minimum service time of processes is small for seed 2, which may be why FCFS has the largest waiting time in seed 1 but not in seed 2.)



```
Using seed: 1797410758
Processes to be executed:
    [#0]: State: ProcessStates.NEW, Arrival: 0.01874985913506718, Service: 0.024082859896149694, Remaining: 0.024082859896149694
    [#1]: State: ProcessStates.NEW, Arrival: 0.34659969422148434, Service: 0.04939805767338781, Remaining: 0.04939805767338781
    [#2]: State: ProcessStates.NEW, Arrival: 1.0702461107834187, Service: 8.589090636275131, Remaining: 8.589090636275131
    [#3]: State: ProcessStates.NEW, Arrival: 1.082379556436702, Service: 2.893534830133524, Remaining: 2.893534830133524
    [#4]: State: ProcessStates.NEW, Arrival: 1.1636907225178434, Service: 0.20712579697293265, Remaining: 0.20712579697293265
    [#5]: State: ProcessStates.NEW, Arrival: 1.1763155340637397, Service: 0.12798104125124352, Remaining: 0.12798104125124352
    [#6]: State: ProcessStates.NEW, Arrival: 1.372802000843104, Service: 0.2428342995239517, Remaining: 0.2428342995239517
    [#7]: State: ProcessStates.NEW, Arrival: 1.9233807328523982, Service: 2.7156449945284207, Remaining: 2.7156449945284207
    [#8]: State: ProcessStates.NEW, Arrival: 2.7497838693531627, Service: 1.225013439022086, Remaining: 1.225013439022086
    [#9]: State: ProcessStates.NEW, Arrival: 3.408243345957417, Service: 4.955566062642633, Remaining: 4.955566062642633
```

This may be the reason why for the first seed, the non-preemptive scheduling algorithms SJF and FCFS have larger waiting time than the preemptive algorithms SRTF and RR, and since SRTF has provable minimum average waiting time, the waiting time of SRTF is less than that of RR, and for the same reason, the waiting time of SJF is less than that of FCFS.
Here, in the same way as for the second seed, in RR - while P1 executes the second time, 4 processes (P4,P5,P6,P7) arrive, and therefore P1 has to be added to the end of queue after all these 4 processes once its time slice finishes, and thus the waiting time will be high - however the increase in waiting time from this is less than the high waiting time for the non preemptive SJF and FCFS due to large service times of processes.

2. **Second seed - deviation and reason of deviation**

The service times may also explain the reason why the second seed also shows the average waiting times that deviated from our expected order. For the second seed the average waiting times (and the average turnaround times) of the scheduling algorithms were in the order of: SRTF < SJF < FCFS < RR. The average waiting time of SRTF < SJF was as expected, but the average waiting time of FCFS < RR was not as expected as we expected the two to be in the opposite order.
This may be due to the fact that the second seed has the smallest total service time for the processes in the queue, of 17.6355108934645539, and therefore the processes were least affected by the problem of starvation in the non-preemptive algorithms SJF and FCFS and this may have lead to SJF and FCFS having lower waiting times than RR; since there are a lot of (10) different processes and each process can only be executed for the time slice of 0.5 each time for RR, the waiting time of each process in RR could be quite long. For example, from the Gantt chart, P0 executes then P1, then P0 executes, then P1, then P2 then P3 and so on - while P1 executes the second time, 4 processes arrive (P4,P5,P6 and P7) - this means P1 has to be added to the end of the queue after all these 4 processes earlier in queue once the time slice of P1 is over. Thus at the end when P1 has only a little remaining time left, P1 still has to wait for many processes to finish earlier in the queue; thus this gives a high average waiting time for RR which is greater than that of SJF and FCFS in which processes only have to wait for processes earlier in queue of small service times to finish.