CSEN383 - Project 2

Submission Details:

Group: 6

Class: CSEN 383-1 Advanced Operating Systems

Objective:
This assignment will give you experience with process scheduling algorithms. Write a Java or C program that performs runs for fixed (same) workload using each of the following process scheduling algorithms:

- **First-come first-served (FCFS) // non-preemptive**
- **Shortest job first (SJF) // non-preemptive**
- **Shortest remaining time (SRT) // preemptive**
- **Round robin (RR) // preemptive**
- **Highest priority first (HPF) // both non-preemptive and preemptive**

We evaluated 6 classic CPU scheduling algorithms - FCFS, SJF, SRT, RR, HPF (non-preemptive and preemptive along with each) over 5 independent iterations. We report averages for 4 statistics: Turnaround time, waiting time, response time, throughput. Results align with theoretical expectations with nuances driven by preemption overheads and the workload's distribution of burst times and priorities.

Requirements:
Run each algorithm for 100 quanta (time slices), labeled 0 through 99. Before each run, generate a set of simulated processes. For each simulated process, randomly generate:

- An arrival time: an integer value from 0 through 99 (measured in quanta).
- An expected total run time: an integer value from 1 through 10 quanta.
- A priority: integer 1, 2, 3, or 4 (1 is highest)

Conclusion by metric:
The output result

	Avg. Turnaround	Avg. Waiting	Avg. Response	Throughput
FCFS	35.21	29.56	29.56	0.18
SJF	14.27	8.84	8.84	0.18
SRT	9.69	6.20	27.7	0.29
RR	28.60	25.59	7.74	0.13
HPF-Np	35.12	29.18	29.18	0.17
HPF-P	35.56	29.90	29.25	0.18

Avg. Turnaround Time:

SRT delivers the **lowest turnaround time (9.69)** by continuously selecting the job with the shortest remaining time, minimizing delays for short processes. SJF also performs well (14.27) since it picks short jobs early but lacks preemption, making it slightly worse than SRT when short jobs arrive after a long one has started. FCFS (35.21), HPF-Np (35.12), and HPF-P (35.56) perform poorly because they may leave short jobs waiting behind long ones. RR (28.60) does better than FCFS but still lags behind SRT and SJF because its time slicing introduces context-switching overhead and does not minimize job completion time.

Avg. Waiting Time:

SRT again comes out best with the **shortest average waiting time (6.20)**, as preemption prevents short jobs from sitting idle while long ones run. SJF follows at 8.84, benefiting from picking short jobs early but still unable to interrupt longer jobs in progress. RR (25.59) has lower waiting than FCFS (29.56) and HPF variants (~29), thanks to its round-robin fairness, but short jobs still accumulate some wait across multiple time slices. FCFS and HPF cause long queues for later arrivals, keeping waiting time high.

Avg. Response Time:

RR achieves the **best response time (7.74)** because every process gets a CPU slice quickly after arrival, regardless of burst length. This is particularly advantageous for interactive jobs. SJF (8.84) and SRT (27.7) have weaker response times—SJF gives immediate service to short jobs but long ones wait; SRT prioritizes shortest remaining time, which can delay large processes

significantly if short jobs keep arriving. FCFS (29.56) and HPF (around 29) perform worst because long jobs block later ones entirely, leading to high initial response delays.

Throughput:

SRT achieves the **highest throughput (0.29)**, completing more jobs per unit time by prioritizing shorter tasks and maximizing completions. SJF and the priority algorithms (0.17–0.18) are roughly tied, hindered when long jobs block the queue. FCFS and HPF also stay low at ~0.18 for similar reasons. RR (0.13) has the lowest throughput because frequent context switches and fixed time slices slow down overall job completion rates.