RIGHT PAGE

CYCLE 1
(OPERATING SYSTEMS LAB)
Experiment 1.1

Date:

CPU Scheduling Algorithms

Aim: Implement the following four CPU scheduling algorithms and compute the Completion Time, Turnaround Time and Waiting Time of each process, Average Waiting Time and Average Turnaround Time.

1. First-Come-First-Serve (FCFS / FIFO)
2. Shortest Job First (SJF) – non-preemptive
3. Round Robin – preemptive (with a user-defined time quantum)
4. Priority Scheduling – preemptive or non-preemptive

Algorithm:

Algorithm_FCFS:

1. Sort processes by arrival time (if needed; in the code, input order is assumed to be arrival order).
2. Initialize `time = 0`.
3. For each process in order:
    - If `time < arrival_time`, update `time = arrival_time`.
    - Execute the process for its burst time.
    - Set `completion_time = time + burst_time`.
    - Calculate `turnaround_time = completion_time - arrival_time`.
    - Calculate `waiting_time = turnaround_time - burst_time`.
4. Print results and calculate averages.

Algorithm_SJF:

1. Initialize `time = 0`, and mark all processes as not completed.
2. Repeat until all processes are done:
    - Among the arrived and not completed processes, choose the one with the shortest burst time.
    - If no process has arrived yet, increment `time++`.
    - Else:
        - Execute the selected process to completion.
        - Update `completion_time`, `turnaround_time`, `waiting_time`.
        - Mark the process as done.
3. Print results and calculate averages.

RIGHT PAGE

Algorithm_RR:

1. Input the time quantum.
2. Initialize a queue and track whether each process is already in the queue.
3. Start from `time = 0`. Enqueue all processes that have arrived by current time.
4. While all processes are not completed:
    - If the queue is empty, increment `time++`.
    - Else:
        - Dequeue a process.
        - If its remaining time ≤ quantum:
            - Run it till completion.
            - Set `completion_time = current_time`.
        - Else:
            - Run it for `quantum` time.
            - Decrease `remaining_time`.
            - Re-enqueue it if it still has time left.
        - Enqueue any new processes that have arrived by now and are not yet in queue.
5. Once done, calculate `turnaround_time`, `waiting_time`.
6. Print results.

Algorithm_Priority:

1. Start from `time = 0`.
2. While not all processes are complete:
    - From all processes that have arrived and not completed, select the one with the highest priority (i.e., lowest number).
    - If no process is available, increment `time++`.
    - Else:
        - Run the selected process for 1 time unit.
        - Decrease its `remaining_time`.
        - If it becomes 0, set `completion_time`, mark as done.
3. Once all are complete, compute `turnaround_time`, `waiting_time`.
4. Print results and averages.

Result: Implemented FCFS, SJF, Priority, and Round Robin scheduling. Computed CT, TAT, WT for each and compared average WT and average TAT across algorithms.

LEFT PAGE

Sample input:

processes	Arrival time	Burst time	priority
p1	0	5	2
p2	1	3	1
p3	2	8	4
p4	3	6	3

Output and Observations:

FCFS:

processes	Arrival time	Burst time	priority	CT	TAT	WT
p1	0	5	2			
p2	1	3	1			
p3	2	8	4			
p4	3	6	3			

Gantt Chart:

Average TAT:
Average WT:

SJF:

processes	Arrival time	Burst time	priority	CT	TAT	WT
p1	0	5	2			
p2	1	3	1			
p3	2	8	4			
p4	3	6	3			

Gantt Chart:

Average TAT:
Average WT:

Round Robin:

processes	Arrival time	Burst time	priority	CT	TAT	WT
p1	0	5	2			
p2	1	3	1			
p3	2	8	4			
p4	3	6	3			

Gantt Chart:

Average TAT:
Average WT:

Priority:

processes	Arrival time	Burst time	priority	CT	TAT	WT
p1	0	5	2			
p2	1	3	1			
p3	2	8	4			
p4	3	6	3			

Gantt Chart:

Average TAT:
Average WT: