Practical No 8

Aim: CPU Scheduling Algorithm (Part 2)-Round Robin

8.1 Implement Round Robin scheduling with configurable time quantum.
8.2 Compare with FCFS: fairness, turnaround, response time.
8.3 rack context switches and improve queue management.

```
from collections import deque

def roundRobin(processes, timeQuantum):
    ""
    processes: list of tuples (pid, arrivalTime, burstTime)
    timeQuantum: int
    """
    n = len(processes)
    processes.sort(key=lambda x: x[1])  # sort by arrival time

    remainingBt = {pid: bt for pid, at, bt in processes}
    completionTime = {}
    turnaroundTime = {}
    waitingTime = {}
    responseTime = {}
    startTime = {}
    ganttChart = []
    readyQueue = deque()
    currentTime = 0
    visited = [False] * n
    completed = 0
    contextSwitches = 0
    prevPid = None

    while completed < n:
        # Add processes that have arrived
        for i, (pid, at, bt) in enumerate(processes):
            if at <= currentTime and not visited[i]:
                readyQueue.append(pid)
                visited[i] = True
        if readyQueue:
            pid = readyQueue.popleft()
            if pid != prevPid and prevPid is not None:
                contextSwitches += 1
            prevPid = pid

            if pid not in startTime:
                startTime[pid] = currentTime
                responseTime[pid] = currentTime - next(at for p, at, bt in processes if p == pid)
            execTime = min(timeQuantum, remainingBt[pid])
            ganttChart.append((pid, currentTime, currentTime + execTime))
            currentTime += execTime
            remainingBt[pid] -= execTime
            # Add new arrivals during execution
            for i, (p, at, bt) in enumerate(processes):
                if at <= currentTime and not visited[i] and p not in readyQueue:
                    readyQueue.append(p)
                    visited[i] = True
            if remainingBt[pid] > 0:
                readyQueue.append(pid)
            else:
                completionTime[pid] = currentTime
                completed += 1
        else:
            currentTime += 1  # CPU idle

    # Calculate TAT & WT
    for pid, at, bt in processes:
        turnaroundTime[pid] = completionTime[pid] - at
        waitingTime[pid] = turnaroundTime[pid] – bt
    avgWt = sum(waitingTime.values()) / n
    avgTat = sum(turnaroundTime.values()) / n
    avgRt = sum(responseTime.values()) / n
    # Print results
    print("\n--- Round Robin Scheduling ---")
    print(f"Time Quantum: {timeQuantum}")
    print("PID\tAT\tBT\tST\tCT\tTAT\tWT\tRT")
    for pid, at, bt in processes:
        print(f"{pid}\t{at}\t{bt}\t{startTime[pid]}\t{completionTime[pid]}\t"
              f"{turnaroundTime[pid]}\t{waitingTime[pid]}\t{responseTime[pid]}")
    print(f"\nAverage Waiting Time: {avgWt:.2f}")
    print(f"Average Turnaround Time: {avgTat:.2f}")
    print(f"Average Response Time: {avgRt:.2f}")
    print(f"Context Switches: {contextSwitches}")
    # Print Gantt Chart
    print("\nGantt Chart:")
    for pid, start, end in ganttChart:
        print(f"| P{pid} ({start}-{end}) ", end="")
    print("|")

def fcfs(processes):
    processes.sort(key=lambda x: x[1])  # sort by arrival time
    n = len(processes)
    currentTime = 0
    startTime = {}
    completionTime = {}
    turnaroundTime = {}
    waitingTime = {}
    responseTime = {}
    ganttChart = []
    for pid, at, bt in processes:
        if currentTime < at:
            currentTime = at
        startTime[pid] = currentTime
        responseTime[pid] = currentTime - at
        ganttChart.append((pid, currentTime, currentTime + bt))
        currentTime += bt
        completionTime[pid] = currentTime
        turnaroundTime[pid] = completionTime[pid] - at
        waitingTime[pid] = turnaroundTime[pid] - bt
    avgWt = sum(waitingTime.values()) / n
    avgTat = sum(turnaroundTime.values()) / n
    avgRt = sum(responseTime.values()) / n
    # Print results
    print("\n--- First-Come First-Serve (FCFS) ---")
    print("PID\tAT\tBT\tST\tCT\tTAT\tWT\tRT")
    for pid, at, bt in processes:
        print(f"{pid}\t{at}\t{bt}\t{startTime[pid]}\t{completionTime[pid]}\t"
              f"{turnaroundTime[pid]}\t{waitingTime[pid]}\t{responseTime[pid]}")
    print(f"\nAverage Waiting Time: {avgWt:.2f}")
    print(f"Average Turnaround Time: {avgTat:.2f}")
    print(f"Average Response Time: {avgRt:.2f}")
    print("Context Switches: N/A (no preemption)")
    # Print Gantt Chart
    print("\nGantt Chart:")
    for pid, start, end in ganttChart:
        print(f"| P{pid} ({start}-{end}) ", end="")
    print("|")
# Example usage
processList = [
    (1, 0, 5),
    (2, 1, 4),
    (3, 2, 2),
    (4, 4, 1)
]
fcfs(processList.copy())
roundRobin(processList.copy(), timeQuantum=2)
```

```
--- First-Come First-Serve (FCFS) ---
PID     AT      BT      ST      CT      TAT     WT      RT
1       0       5       0       5       5       0       0
2       1       4       5       9       8       4       4
3       2       2       9       11      9       7       7
4       4       1       11      12      8       7       7
Average Waiting Time: 4.50
Average Turnaround Time: 7.50
Average Response Time: 4.50
Context Switches: N/A (no preemption)
Gantt Chart:
| P1 (0-5) | P2 (5-9) | P3 (9-11) | P4 (11-12) |
--- Round Robin Scheduling ---
Time Quantum: 2
PID     AT      BT      ST      CT      TAT     WT      RT
1       0       5       0       12      12      7       0
2       1       4       2       11      10      6       1
3       2       2       4       6       4       2       2
4       4       1       8       9       5       4       4
Average Waiting Time: 4.75
Average Turnaround Time: 7.75
Average Response Time: 1.75
Context Switches: 6
Gantt Chart:
| P1 (0-2) | P2 (2-4) | P3 (4-6) | P1 (6-8) | P4 (8-9) | P2 (9-11) | P1 (11-12) |
```