Part E

① Consider the following processes with arrival times and burst times.

Process	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	6

~~P4~~

Calculate the average waiting time using First-Come First-Served (FCFS) scheduling.

⟹ Step 1: Compute Completion Time (CT)

Process	Arrival Time (AT)	Burst Time (BT)	Completion Time (CT)
P1	0	5	0+5=5
P2	1	3	5+3=8
P3	2	6	8+6=14

Step 2: Compute Turnaround Time (TAT)

TAT = CT - AT

Process	Arrival Time ~~(AT)~~	Completion Time (CT)	Turnaround Time (TAT)
P1	0	5	5-0=5
P2	1	8	8-1=7
P3	2	14	14-2=12

Step 3: Compute Waiting Time (WT)

WT = TAT - BT

Process	Turnaround Time (TAT)	Burst Time (BT)	Waiting time (WT)
P_1	5	5	5-5 = 0
P_2	7	3	7-3 = 4
P3	12	6	12-6 = 6

Step 4: Compute Average Waiting Time (AWT)

$$AWT = \frac{\Sigma WT}{\text{Total processes}}$$

$$AWT = \frac{0+4+6}{3}$$

$$= \frac{10}{3}$$

$$\boxed{AWT = 3.33.}$$

(2) Consider the following processes with arrival times and burst times:

Process	Arrival time	Burst Time
P1	0	3
P2	1	5
P3	2	1
P4	3	4

Calculate the average turnaround time using shortest job first (SJF) scheduling.

⇒ Step 1:

Process	Arrival Time (AT)	Burst Time (BT)
P1	0	3
P2	1	5
P3	2	1
P4	3	4

Step 2: Determine the execution order and completion time.

1) At Time 0, only P1 is available → Execute P1 (BT = 3).

Completion Time (CT) of P1 = 0 + 3 = 3

2) At Time 3, the available processes are P2 (BT = 5), P3 (BT = 1), P4 (BT = 4)

Shortest Burst Time = P3 (BT = 1) → execute P3.

Completion time (CT) of P3 = 3 + 1 = 4

3) At Time 4, the available processes are P2 (BT = 5), P4 (BT = 4).

Shortest Burst Time = P4 (BT = 4) → execute P4

Completion Time (CT) of P4 = 4 + 4 = 8

4) At time 8, only P2 remains → Execute P2 (BT = 5).
Completion Time (CT) of P2 = 8 + 5 = 13.

The execution order is P1 → P3 → P4 → P2.

Step 3: Compute Turnaround Time (TAT)

TAT = CT − AT

Process	Arrival Time (AT)	Completion Time (CT)	Turnaround Time (TAT)
P1	0	3	3 − 0 = 3
P2	1	13	13 − 1 = 12
P3	2	4	4 − 2 = 2
P4	3	8	8 − 3 = 5

Step 4: Compute Average Turnaround Time (ATAT)

$$ATAT = \frac{\Sigma TAT}{\text{total processes}}.$$

$$ATAT = \frac{3+12+2+5}{4}$$

$$= \frac{22}{4}$$

$$\boxed{ATAT = 5.5 \text{ ms.}}$$

(3) Consider the following processes with arrival times burst times, and priorities (lower number indicates higher priority).

Process	Arrival Time	Burst Time	priority
P1	0	6	3
P2	1	4	1
P3	2	7	4
P4	3	2	2

Calculate the average waiting time using priority scheduling.

⇒ Step1: sorted processes by Arrival time and execute based on priority

Process	Arrival Time (AT)	Burst Time (BT)	Priority
P_1	0	6	3
P_2	1	4	1
P_3	2	7	4
P_4	3	2	2

Step2: Determination execution order and completion time (CT):

1) At the time 0, only P1 is available → execute P1 (BT=6)
   Completion time (CT) of P1 = 0+6 = 6

2) At the time 6, the available processes are P2 (Priority=1) P3 (priority = 4), P4 (priority = 2).
   Highest priority (lowest ~~priority~~ number) = P2 (priority = 1) ⇒ execute P2 (BT=4)
   Completion time (CT) of P2 = 6+4 = 10.

3) At the time 10, the available processes are P3 (priority=4) and P4 (Priority=2).

Highest Priority = P4 (Priority = 2) → execute P4 (BT=2).

Completion time (CT) of P4 = 10 + 2 = 12

4) At time 12, only P3 remains → execute P3 (BT=7)

Completion Time (CT) of P3 = 12 + 7 = 19

Execution order is P1 → P2 → P4 → P3.

Step 3: Compute turnaround time (TAT)

TAT = CT - BT

Process	Arrival Time (AT)	Completion Time (CT)	Turnaround Time (TAT)
P1	0	6	6 - 0 = 6
P2	1	10	10 - 1 = 9
P3	2	19	19 - 2 = 17
P4	3	12	12 - 3 = 9

Step 4: Compute Waiting Time (WT)

WT = TAT - BT

Process	Turnaround Time (TAT)	Burst Time (BT)	Waiting Time (WT)
P1	6	6	6 - 6 = 0
P2	9	4	9 - 4 = 5
P3	17	7	17 - 7 = 10
P4	9	2	9 - 2 = 7

Step 5: Compute Average Waiting time (AWT)

$$AWT = \frac{\Sigma WT}{\text{Total processes}} = \frac{0+5+10+7}{4} = \frac{22}{4}$$

$$\boxed{AWT = 5.5 \text{ ms}}$$

(4) Consider the following processes with arrival times and burst times, and the time quantum for Round Robin Scheduling is 2 units:

Process	Arrival Time	Burst Time
P1	0	4
P2	1	5
P3	2	2
P4	3	3

Calculate the average turnaround time using Round Robin Scheduling.

⟹

Step 1: Given Data

Process	Arrival Time	Burst Time
P1	0	4.
P2	1	5
P3	2	2
P4	3	3.

Time quantum = 2 units.

Step 2: Gantt Chart:

1) At time = 0, P1 Starts execution for 2 units (Remaining burst time: 2).

Gantt chart : [P1 (0-2)]

2) At time = 2, P2 arrives and starts execution for 2 units (Remaining burst time: 3)

Gantt Chart: [P1(0-2), P2 (2-4)]

3) At time = 4, P3 arrives and executes for 2 units (completed)

Gantt chart: [P1 (0-2), P2 (2-4) P3 (4-6)]

4) At time = 6, P4 arrives and executes for 2 units (Remaining burst time 1)

Gantt Chart :[P1 (0-2), P2 (2-4), P3(4-6), P4 (6-8)]

5) At time = 8, P1 resumes execution for 2 units (completed)

Gantt chart : [P1(0-2), P2 (2-4), P3 (4-6), P4 (6-8), P1 (8-10)]

6) At time = 10, P2 resumes execution for 2 units (remaining burst time : 1)

Gantt chart : [P1(0-2), P2(2-4), P3(4-6), P4 (6-8), P1(8-10), P2 (10-12)]

7) At time = 12, P4 resumes execution for 1 unit (completed)

Gantt chart : [P1 (0-2), P2 (2-4), P3 (4-6), P4 (6-8), P1 (8-10), P2 (10-12), P4 (12-13)]

8) At time = 13, P2 resumes execution for 1 unit (completed)

Gantt chart : [P1 (0-2), P2(2-4), P3 (4-6), P4 (6-8), P1 (8-10), P2 (10-12), P4 (12-13), P2 (13-14)]

Step 3: Completion time (CT) calculation.

Process	Completion Time (CT)
P1	10
P2	14
P3	6
P4	13

Step 4: Turnaround time (TAT) calculation

TAT = CT - Arrival Time.

Process	Arrival time (AT)	Completion Time (CT)	Turnaroun Time (TAT)
P1	0	10	10 - 0 = 10
P2	1	14	14 - 1 = 13
P3	2	6	6 - 2 = 4
P4	3	13	13 - 3 = 10

Step 5: Average Turnaround Time Calculation

$$\text{Average TAT} = \frac{\Sigma TAT}{\text{Total processes}}$$

$$= \frac{10 + 13 + 4 + 10}{4}$$

$$= \frac{37}{4}$$

$$\boxed{\text{ATAT} = 9.25 \text{ units.}}$$

(5) consider a program that uses the fork() system call to create a child process, initially the parent process has a variable x with a value of 5. After forking, both the parent and child processes increment the value of x by 1. What will be the final values of x in the parent and child processes after the fork() call?

⟹ step by step execution:

1) Initial condition:

The parent process has a variable x = 5.

2) Forking:

- The fork() call creates a new child process.
- Both the parent and child inherit the same initial state, meaning x = 5 in both processes.
- However, each process gets its own separate copy of x.

3) Incrementing x:

- The process of parent increments its copy of x → x = 6.
- The child process also increments its own copy of x → x = 6.

Final values of x:

- In the parent process x = 6
- In the child process x = 6.