1.

process	Arrival Time	Burst Time
P1	0	5
P2	1	7
P3	3	4

Given data :-

* Time quantum : 2 units
* processes :-

P1 : Arrival Time = 0, Burst Time = 5

P2 : Arrival Time = 1, Burst Time = 7

P3 : Arrival Time = 3, Burst Time = 4

① Time 0-2; P, executes for 2 units

Remaining burst times :-

* $P_1 : 3$
* $P_2 : 7$
* $P_3 : 4$

② Time 2-4 : P_2 executes for 2 units

Remaining burst times :-

* $P_1 : 3$
* $P_2 : 5$
* $P_3 : 4$

③ Time 4-6 : P_3 executes for 2 units

Remaining burst times :-

* $P_1 : 3$
* $P_2 : 5$
* $P_3 : 2$

4. Time 6-8:- P_1 executes for 2 units
Remaining burst time
- P_1 : 1
- P_2 : 5
- P_3 : 2

5. Time 8-10:- P_2 executes for 2 units
Remaining burst times:-
- P_1 : 1
- P_2 : 3
- P_3 : 2

6. Time 10-12 : P_3 executes for 2 units
Remaining burst times:
- P_1 : 1
- P_2 : 3
- P_3 :- 0 (P_3 completes at time 12)

7. Time 12-13 : P_1 executes for 1 units:
Remaining burst times:
- P_1 : 0
- P_2 : 3

8. Time 13-15 : P_2 executes for 2 units:
Remaining burst times:-
- P_1 : 0
- P_2 : 1

9. Time 15-16: P_2 executes for 1 unit
Remaining burst times:
- P_2 : 0

Completion order:

① P_3 (at time 12)
② P_1 (at time 13
③ P_2 (at time 16)

———— × ————

2.

process	Burst time	priority
P_1	8	3
P_2	1	1
P_3	2	5
P_4	1	4

* Arrival Time :- All processes arrive at 0 ms
* priority Rule:- Lower priority number = higher priority

(a) Non-preemptive priority scheduling:-

* In non-preemptive priority scheduling, the CPU executes the process with the highest priority without interruption until it completes.
* Execution order (by priority): $P_2 \rightarrow P_1 \rightarrow P_4 \rightarrow P_3$.

P2	P1	P4	P3
0	1	9	10	12.

(ii) preemptive priority scheduling:-

* At 0 ms, P_2 starts.
* At 1 ms, P_2 completes; P_1 starts.
* NO interruption occur as no higher-priority processes arrive.

- After P_1 completes, P_4 (priority) runs, followed by P_3

$|_0 P_2 |_1 P_1 \quad |_9 P_4 |_{10} P_3 |_{12}$

- Turn around Time = completion Time - Arrival Time
- waiting Time = Turn around Time - Burst Time.

a. Non-preemptive Scheduling:-

process	Burst Time	priority	Completion Time	Turn around Time	waiting Time
P_2	1	1	1	1-0=1	1-1=0
P_1	8	3	9	9-0=9	9-8=1
P_4	1	4	10	10-0=10	10-1=9
P_3	2	5	12	12-0=12	12-2=10

- Average. AT = (1+9+10+12)/4 = 8ms
- Average. wT = (0+1+9+10)/4 = 5ms

b. preemptive scheduling:-

process	Burst Time	priority	completion Time	Turnaround Time	waiting Time
P_2	1	1	1	1-0=1	1-1=0
P_1	8	3	9	9-0=9	9-8=1
P_4	1	4	10	10-0=10	10-1=9
P_3	2	5	12	12-0=12	12-2=10

* Average. TAT = 8 ms
* Average. WT = 5 ms

① Gantt charts:-

* Non-preemptive: |P2|P1|P4|P3|
* preemptive: Same as non-preemptive in this case.

2. Average Times:-

* Average TAT = 8 ms
* Average WT = 5 ms

3. P_1:-

$P_1 \rightarrow R_2$

$P_2 \rightarrow R_1$

$P_3 \rightarrow R_1$

$P_4 \rightarrow R_2$

1. P_1:-

* Allocates R_2 → Add edge $R_2 \rightarrow P_1$
* Requests R_1 → Add edge $P_1 \rightarrow R_1$

2. P_2:-

* Allocates R_1 → Add edge $R_1 \rightarrow P_2$

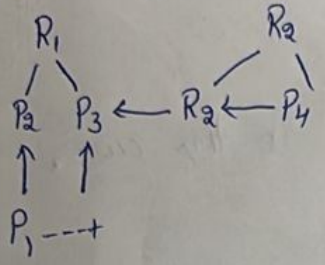
3. P_3:-

* Allocates R_1 → Add edge $R_1 \rightarrow P_3$
* Requests R_2 → Add edge $P_3 \rightarrow R_2$

4. P_4:-

* Allocates $R_2 \rightarrow$ Add edge $R_2 \rightarrow P_4$

Graph Layout:-



* Cycle detection:-

* $P_1 \rightarrow R_1 \rightarrow P_3 \rightarrow R_2 \rightarrow P_1$

* This forms cycle :. $P_1 \rightarrow R_1 \rightarrow R_3 \rightarrow R_2 \rightarrow P_1$

(k) Deadlock Analysis:-

Deadlock conditions:-

1. There is a cycle in RAG

2. All process in the cycle are waiting for resources that cannot be allocated due to unavailable.

P_1:- Allocates R_2 & waits for R_1

P_3:- Allocates R_1 & waits for R_2

Both P_1 and P_3 are part of the cycle & Cannot proceed

∴ Hence the System is in deadlock.//

4. Analyze using Banker's algorithm.

① Total Resources: a tape drives.

② Allocation & Maximum Requirement: |process| current Allocation | Maximum Requirement |

P1	13	17
P2	11	16
P3	13	15

③ Total resources - Allocated resources.

process	maximum_Requirement	current Allocation	Need
P_1	7	3	4
P_2	6	1	5
P_3	5	3	2

① check P_1:-
Need (4) > Available (2) → P_1 cannot be estafied.

② check P_2:-
Need (5) > available (2) → P_2 cannot be satisfied.

③ check P_3:-
Need (2) ≤ Available (2) → P_3 can be satisfied.

* New Available Resources : 2+3 = 5.

Next Iterations: -

④ check P_1:-

Need (4) ≤ Available(5) → P_1 can be satisfied

* New Available Resources: 5+3 = 8

⑤ check P_2:-

Need (5) ≤ Available (8) → P_2 can be satisfied

* New Available Resources: 8+1 = 9 //.