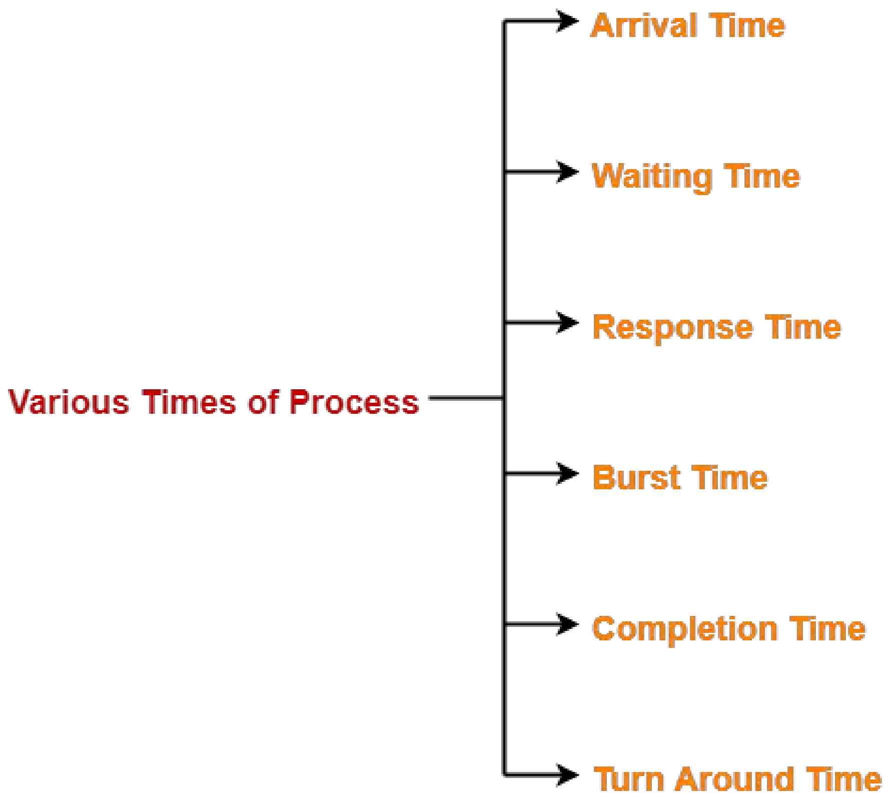
Numericals for practice

Various times related to process are-



1. Arrival Time-

- Arrival time is the point of time at which a process enters the ready queue.

2. Waiting Time-

- Waiting time is the amount of time spent by a process waiting in the ready queue for getting the CPU.

Waiting time = Turn Around time – Burst time

3. Response Time-

- Response time is the amount of time after which a process gets the CPU for the first time after entering the ready queue.

Response Time = Time at which process first gets the CPU – Arrival time

4. Burst Time-

- Burst time is the amount of time required by a process for executing on CPU.
- It is also called as **execution time** or **running time**.
- Burst time of a process can not be known in advance before executing the process.
- It can be known only after the process has executed.

5. Completion Time-

- Completion time is the point of time at which a process completes its execution on the CPU and takes exit from the system.
- It is also called as **exit time**.

6. Turn Around Time-

- Turn Around time is the total amount of time spent by a process in the system.
- When present in the system, a process is either waiting in the ready queue for getting the CPU or it is executing on the CPU.

Turn Around time = Burst time + Waiting time

OR

Turn Around time = Completion time – Arrival time

<u>Important Note-</u>

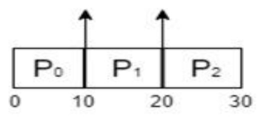
While discussing the above definitions,

- We have considered that the process does not require an I/O operation.
- When process is present in the system, it will be either waiting for the CPU in the ready state or it will be executing on the CPU.

Q.no.1. Consider three CPU-intensive process, which require 10,20 and 30 time units and arrive at times 0,2 and 6 respectively. How many context switches are needed if the operating system implements a shortest remaining time first scheduling algorithm? Do not count the context switches at time zero and at the end.

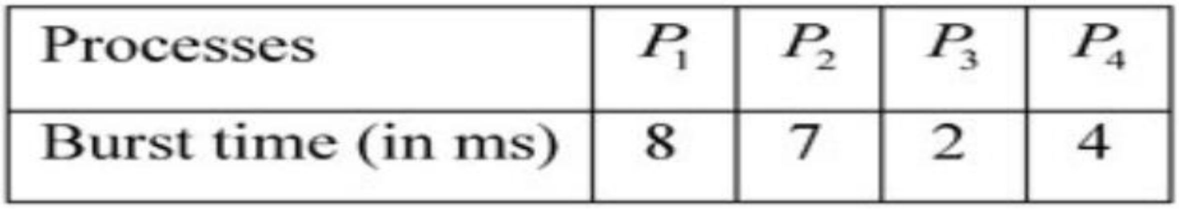
Ans: 2

Explanation



Here is total 4 context switches present but in the question it says do not consider first and last context switches.

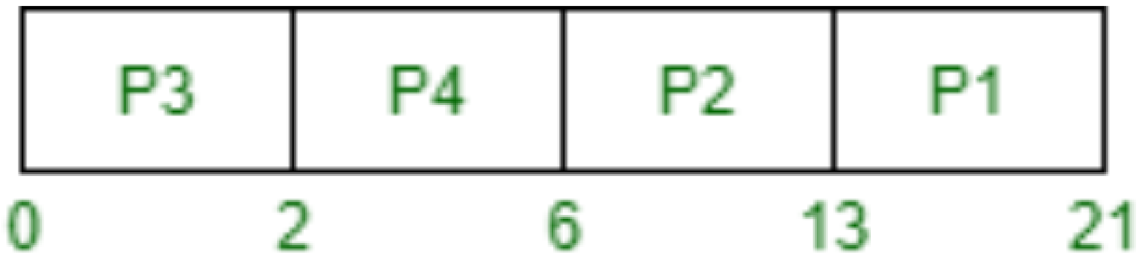
Q.No.2. Consider the following set of processes, assumed to have arrived at time 0. Consider the CPU scheduling algorithms Shortest Job First (SJF) and Round Robin (RR). For RR, assume that the processes are scheduled in the order P_1, P_2, P_3, P_4.



Processes	P_1	P_2	P_3	P_4
Burst time (in ms)	8	7	2	4

If the time quantum for RR is 4 ms, then the absolute value of the difference between the average turnaround times (in ms) of SJF and RR (round off to 2 decimal places) is __________.

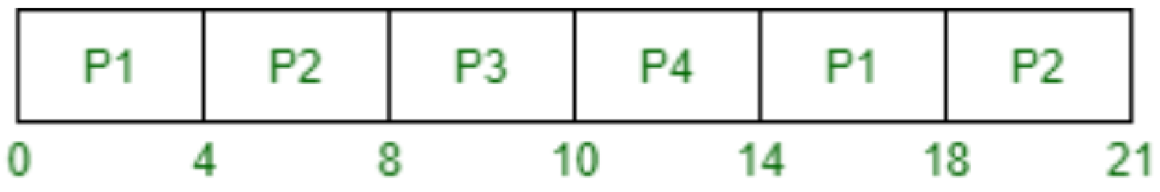
Ans:



Therefore, Average Turn Around Time (TAT) is,

```
= {(21 - 0) + (13 - 0) + (2 - 0) + (6 - 0)} / 4
= 10.5
```

Now, according to Round Robin (RR) CPU Scheduling with time quantum 4, gantt chart is,



Therefore, Average Turn Around Time (TAT) is,

```
= {(18 - 0) + (21 - 0) + (10 - 0) + (14 - 0)} / 4
= 15.75
```

Hence,

```
= |SJF (TAT) - RR(TAT)|
= |10.5 - 15.75|
= 5.25
```

Q.No.3. Consider three process, all arriving at time zero, with total execution time of 10, 20 and 30 units respectively. Each process spends the first 20% of execution time doing I/O, the next 70% of time doing computation, and the last 10% of time doing I/O again. The operating system uses a shortest remaining compute time first scheduling algorithm and schedules a new process either when the running process gets blocked on I/O or when the running process finishes its compute burst. Assume that all I/O operations can be overlapped as much as possible. For what percentage of does the CPU remain idle?

1. 10.6%
2. 30.0%
3. 89.4%
4. 00.0%

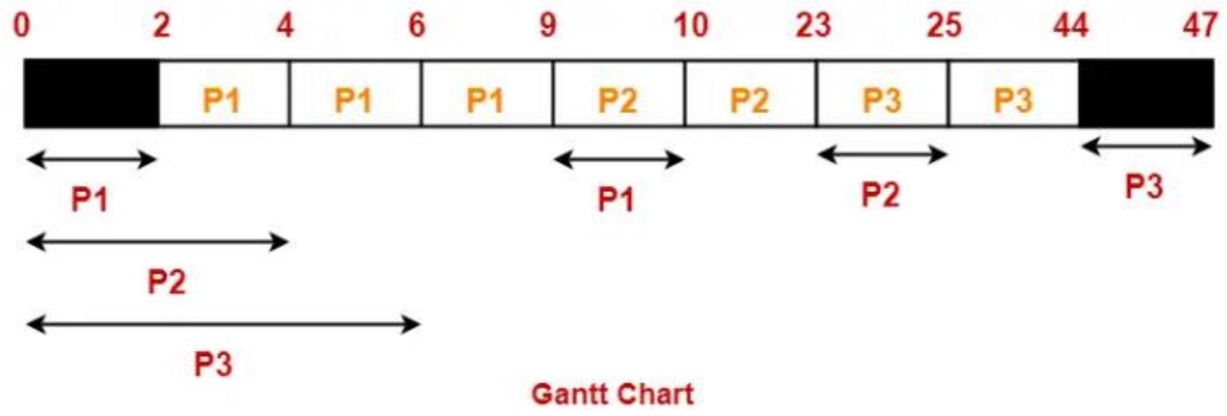
Ans :

According to question, we have-

	Total Burst Time	I/O Burst	CPU Burst	I/O Burst
Process P1	10	2	7	1
Process P2	20	4	14	2
Process P3	30	6	21	3

The scheduling algorithm used is Shortest Remaining Time First.

Gantt Chart-



Percentage of time CPU remains idle

= (5 / 47) x 100

= 10.638%

Thus, Option (1) is correct.

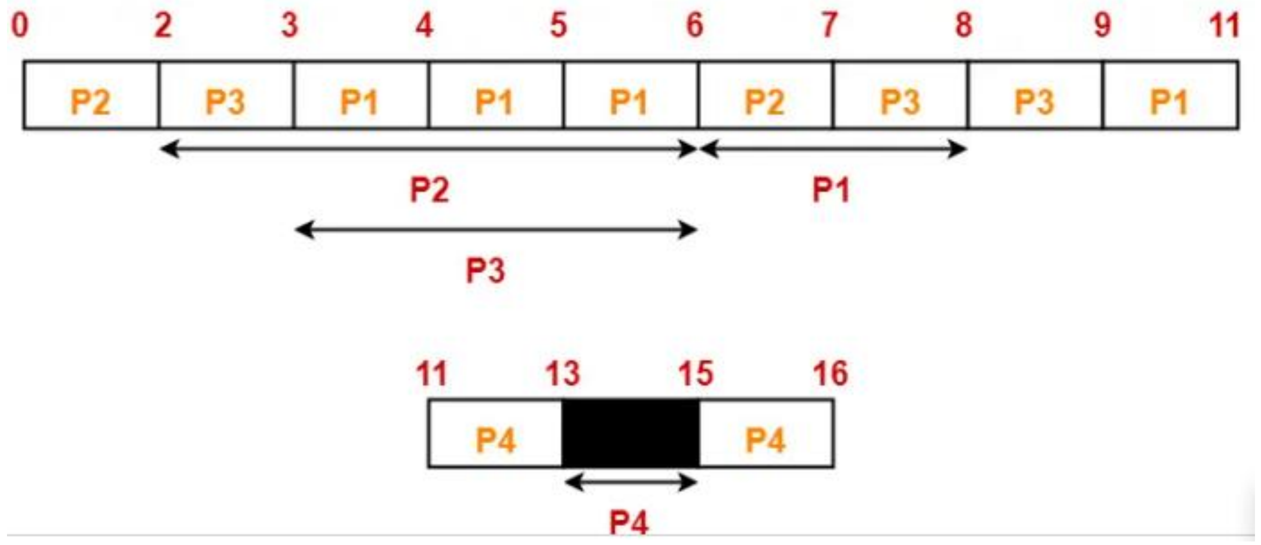
Q.No.4. Consider the set of 4 processes whose arrival time and burst time are given below-

Process No.	Arrival Time	Burst Time		
		CPU Burst	**I/O Burst**	**CPU Burst**
P1	0	3	2	2
P2	0	2	4	1
P3	2	1	3	2
P4	5	2	2	1

If the CPU scheduling policy is Shortest Remaining Time First, calculate the average waiting time and average turn around time.

Ans :

Gantt Chart-



Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	**Exit time**	**Turn Around time**	**Waiting time**
P1	11	11 – 0 = 11	11 – (3+2) = 6
P2	7	7 – 0 = 7	7 – (2+1) = 4
P3	9	9 – 2 = 7	7 – (1+2) = 4
P4	16	16 – 5 = 11	11 – (2+1) = 8

Now,

- Average Turn Around time = (11 + 7 + 7 + 11) / 4 = 36 / 4 = 9 units
- Average waiting time = (6 + 4 + 4 + 8) / 4 = 22 / 5 = 4.4 units

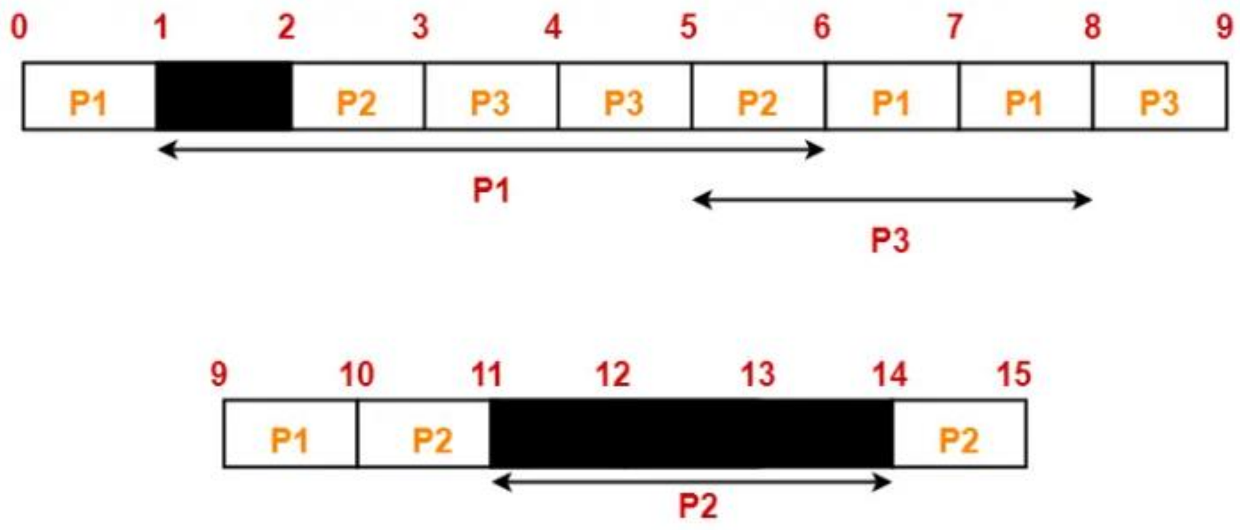
Q.No.5. Consider the set of 4 processes whose arrival time and burst time are given below-

Process No.	**Arrival Time**	**Priority**	**Burst Time**		
			CPU Burst	**I/O Burst**	**CPU Burst**
P1	0	2	1	5	3
P2	2	3	3	3	1
P3	3	1	2	3	1

If the CPU scheduling policy is Priority Scheduling, calculate the average waiting time and average turn around time. (Lower number means higher priority).

Ans :

The scheduling algorithm used is Priority Scheduling.



Gantt Chart

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	10	10 – 0 = 10	10 – (1+3) = 6
P2	15	15 – 2 = 13	13 – (3+1) = 9
P3	9	9 – 3 = 6	6 – (2+1) = 3

Now,

- Average Turn Around time = (10 + 13 + 6) / 3 = 29 / 3 = 9.67 units
- Average waiting time = (6 + 9 + 3) / 3 = 18 / 3 = 6 units

Q.No.6. Consider three processes (process id 0, 1, 2 respectively) with compute time bursts 2, 4 and 8 time units. All processes arrive at time zero. Consider the longest remaining time first (LRTF) scheduling algorithm. In LRTF ties are broken by giving priority to the process with the lowest process id. The average turn around time is:
(A) 13 units
(B) 14 units
(C) 15 units
(D) 16 units

Ans: (A)

Explanation: Let the processes be p0, p1 and p2. These processes will be executed in following order.

```
  p2  p1  p2  p1  p2  p0  p1   p2   p0   p1   p2

0   4   5   6   7   8   9   10    11   12   13   14
```

Turn around time of a process is total time between submission of the process and its completion.
Turn around time of p0 = 12 (12-0)
Turn around time of p1 = 13 (13-0)
Turn around time of p2 = 14 (14-0)

Average turn around time is (12+13+14)/3 = 13.

Q.No.7. Consider the 3 processes, P1, P2 and P3 shown in the table.

Process	Arrival time	Time Units Required
P1	0	5
P2	1	7
P3	3	4

The completion order of the 3 processes under the policies FCFS and RR2 (round robin scheduling with CPU quantum of 2 time units) are
(A)
FCFS: P1, P2, P3

RR2: P1, P2, P3

(B)
FCFS: P1, P3, P2

RR2: P1, P3, P2

(C)
FCFS: P1, P2, P3

RR2: P1, P3, P2

(D)
FCFS: P1, P3, P2

RR2: P1, P2, P3

Ans: (C)

Explanation:
FCFS is clear.

In RR, time slot is of 2 units.

Processes are assigned in following order

p1, p2, p1, p3, p2, p1, p3, p2, p2

This question involves the concept of ready queue. At t=2, p2 starts and p1 is sent to the ready queue and at t=3 p3 arrives so then the job p3 is queued in ready queue after p1. So at t=4, again p1 is executed then p3 is executed for first time at t=6.

Q.No.8. Consider the following table of arrival time and burst time for three processes P0, P1 and P2.

Process	Arrival time	Burst Time
P0	0 ms	9 ms
P1	1 ms	4 ms
P2	2 ms	9 ms

The pre-emptive shortest job first scheduling algorithm is used. Scheduling is carried out only at arrival or completion of processes. What is the average waiting time for the three processes?
(A) 5.0 ms
(B) 4.33 ms
(C) 6.33
(D) 7.33

Ans: (A)

Q.No.9. Which of the following statements are true?

I. Shortest remaining time first scheduling may cause starvation

II. Preemptive scheduling may cause starvation

III. Round robin is better than FCFS in terms of response time

(A) I only
(B) I and III only
(C) II and III only
(D) I, II and III

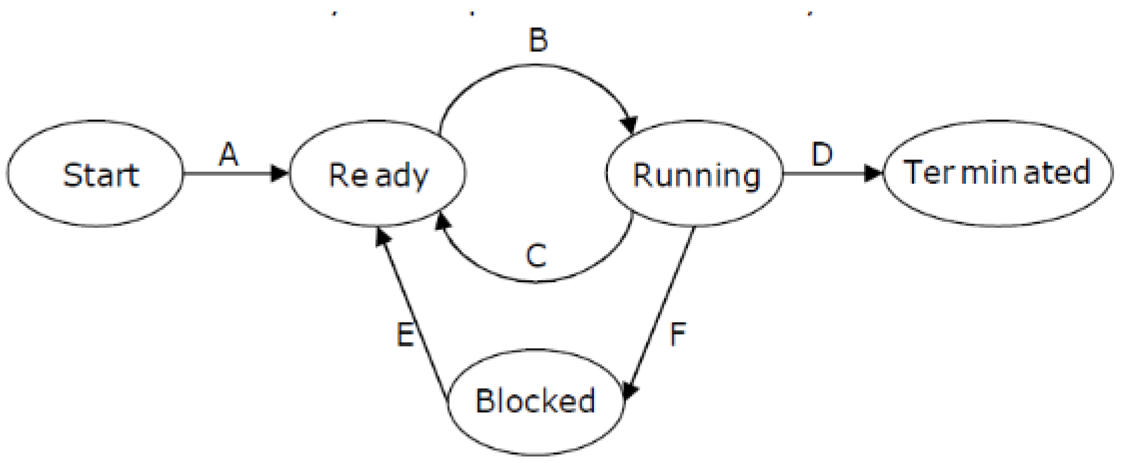
Ans: (D)

Explanation: I) Shortest remaining time first scheduling is a pre-emptive version of shortest job scheduling. In SRTF, job with the shortest CPU burst will be scheduled first. Because of this process, It may cause starvation as shorter processes may keep coming and a long CPU burst process never gets CPU.
II) Pre-emptive just means a process before completing its execution is stopped and other process can start execution. The stopped process can later come back and continue from where it was stopped. In pre-emptive scheduling, suppose process P1 is executing in CPU and after some time process P2 with high priority then P1 will arrive in ready queue then p1 is pre-empted and p2 will brought into CPU for execution. In this way if process which is arriving in ready queue is of higher priority then p1, then p1 is always pre-empted and it may possible that it suffer from starvation.

III) round robin will give better response time then FCFS ,in FCFS when process is executing ,it executed up to its complete burst time, but in round robin it will execute up to time quantum. So Round Robin Scheduling improves response time as all processes get CPU after a specified time.

So, I,II,III are true which is option (D).

Q.No.10. In the following process state transition diagram for a uniprocessor system, assume that there are always some processes in the ready state: Now consider the following statements:



I. If a process makes a transition D, it would result in

another process making transition A immediately.

II. A process P2 in blocked state can make transition E

while another process P1 is in running state.

III. The OS uses preemptive scheduling.

IV. The OS uses non-preemptive scheduling.

Which of the above statements are TRUE?

A I and II

B I and III

C II and III

D II and IV

Ans: (C)

Explanation: I is false. If a process makes a transition D, it would result in another process making transition B, not A.
II is true. A process can move to ready state when I/O completes irrespective of other process being in running state or not.
III is true because there is a transition from running to ready state.
IV is false as the OS uses preemptive scheduling.

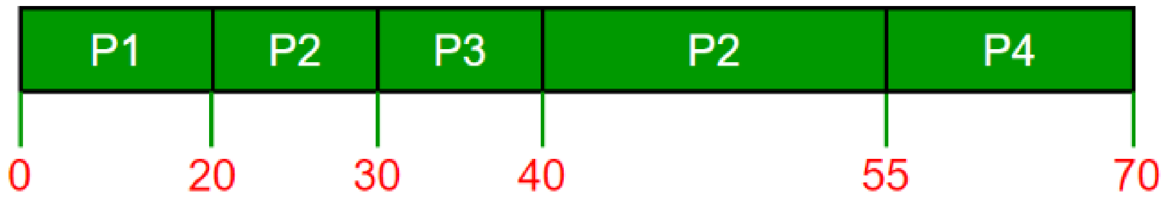
Q.No.11. An operating system uses Shortest Remaining Time first (SRT) process scheduling algorithm. Consider the arrival times and execution times for the following processes:

Process	Execution time	Arrival time
P1	20	0
P2	25	15
P3	10	30
P4	15	45

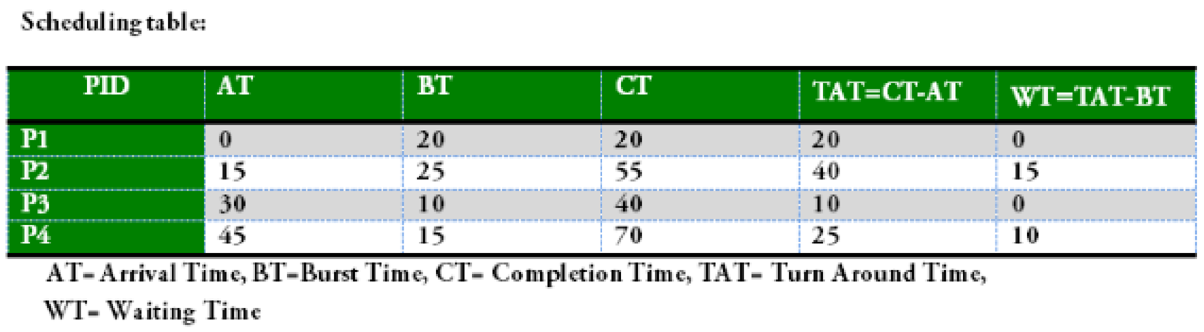
What is the total waiting time for process P2?
(A) 5
(B) 15
(C) 40
(D) 55

Ans: (B)

Explanation: Shortest remaining time, also known as shortest remaining time first (SRTF), is a scheduling method that is a pre-emptive version of shortest job next scheduling. In this scheduling algorithm, the process with the smallest amount of time remaining until completion is selected to execute. Since the currently executing process is the one with the shortest amount of time remaining by definition, and since that time should only reduce as execution progresses, processes will always run until they complete or a new process is added that requires a smaller amount of time.
The Gantt chart of execution of processes:



At time 0, P1 is the only process, P1 runs for 15 time units.
At time 15, P2 arrives, but P1 has the shortest remaining time. So P1 continues for 5 more time units.
At time 20, P2 is the only process. So it runs for 10 time units.
at time 30, P3 is the shortest remaining time process. So it runs for 10 time units.
at time 40, P2 runs as it is the only process. P2 runs for 5 time units.
At time 45, P3 arrives, but P2 has the shortest remaining time. So P2 continues for 10 more time units. P2 completes its execution at time 55.



Scheduling table:

PID	AT	BT	CT	TAT=CT-AT	WT=TAT-BT
P1	0	20	20	20	0
P2	15	25	55	40	15
P3	30	10	40	10	0
P4	45	15	70	25	10

AT- Arrival Time, BT-Burst Time, CT- Completion Time, TAT- Turn Around Time, WT- Waiting Time

As we know, turn around time is total time between submission of the process and its completion. Waiting time is the time The amount of time that is taken by a process in ready queue and waiting time is the difference between Turn around time and burst time.
Total turnaround time for P2 = Completion time – Arrival time
= 55 – 15 = 40
Total Waiting Time for P2= turn around time – Burst time
= 40 – 25 = 15

Q.No.12. Three processes A, B and C each execute a loop of 100 iterations. In each iteration of the loop, a process performs a single computation that requires tc CPU milliseconds and then initiates a single I/O operation that lasts for tio milliseconds. It is assumed that the computer where the processes execute has sufficient number of I/O devices and the OS of the computer assigns different I/O devices to each process. Also, the scheduling overhead of the OS is negligible. The processes have the following characteristics:

Process id	tc	tio
A	100 ms	500 ms
B	350 ms	500 ms
C	200 ms	500 ms

The processes A, B, and C are started at times 0, 5 and 10 milliseconds respectively, in a pure time sharing system (round robin scheduling) that uses a time slice of 50 milliseconds. The time in milliseconds at which process C would complete its first I/O operation is ____________.

(A)500
(B)1000
(C)2000
(D)10000

Ans: (B)

Explanation:
There are three processes A, B and C that run in

round robin manner with time slice of 50 ms.

Processes start at 0, 5 and 10 milliseconds.

The processes are executed in below order

A, B, C, A

50 + 50 + 50 + 50 (200 ms passed)

Now A has completed 100 ms of computations and

goes for I/O now

B, C, B, C, B, C

50 + 50 + 50 + 50 + 50 + 50 (300 ms passed)

C goes for i/o at 500ms and it needs 500ms to

finish the IO.

So C would complete its first IO at 1000 ms

Q.No.13. An operating system uses shortest remaining time first scheduling algorithm for pre-emptive scheduling of processes. Consider the following set of processes with their arrival times and CPU burst times (in milliseconds):

Process	Arrival Time	Burst Time
P1	0	12
P2	2	4
P3	3	6
P4	8	5

The average waiting time (in milliseconds) of the processes is __________.
(A) 4.5
(B) 5.0
(C) 5.5
(D) 6.5

Ans: (C)

Explanation:

Process	Arrival Time	Burst Time
P1	0	12
P2	2	4
P3	3	6

P4	8	5

Burst Time – The total time needed by a process from the CPU for its complete execution.

Waiting Time – How much time processes spend in the ready queue waiting their turn to get on the CPU

Now, The Gantt chart for the above processes is :

P1 - 0 to 2 milliseconds

P2 - 2 to 6 milliseconds

P3 - 6 to 12 milliseconds

P4 - 12 to 17 milliseconds

P1 - 17 to 27 milliseconds

Process p1 arrived at time 0, hence cpu started executing it.

After 2 units of time P2 arrives and burst time of P2 was 4 units, and the remaining time of the process p1 was 10 units,hence cpu started executing P2, putting P1 in waiting state(Pre-emptive and Shortest remaining time first scheduling).

Due to P1's highest remaining time it was executed by the cpu in the end.

Now calculating the waiting time of each process:

P1 -> 17 -2 = 15

P2 -> 0

P3 -> 6 - 3 = 3

P4 -> 12 - 8 = 4

Hence total waiting time of all the processes is

= 15+0+3+4=22

Total no of processes = 4

Average waiting time = 22 / 4 = 5.5

Hence C is the answer.

Q.No.14. Consider the following set of processes, with the arrival times and the CPU-burst times given in milliseconds

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

What is the average turnaround time for these processes with the preemptive shortest remaining processing time first (SRPT) algorithm ?
(A) 5.50
(B) 5.75
(C) 6.00
(D) 6.25

Ans: (A)

Explanation: The following is Gantt Chart of execution

P1	P2	P4	P3	P1
1	4	5	8	12

Turn Around Time = Completion Time – Arrival Time

Avg Turn Around Time = (12 + 3 + 6+ 1)/4 = 5.50

Q.No.15. A uni-processor computer system only has two processes, both of which alternate 10ms CPU bursts with 90ms I/O bursts. Both the processes were created at nearly the same time. The I/O of both processes can proceed in parallel. Which of the following scheduling strategies will result in the least CPU utilization (over a long period of time) for this system ?
(A) First come first served scheduling
(B) Shortest remaining time first scheduling
(C) Static priority scheduling with different priorities for the two processes
(D) Round robin scheduling with a time quantum of 5 ms

Ans: (D)

Explanation: When Round Robin scheduling is used
We are given that the time slice is 5ms. Consider process P and Q.
Say P utilizes 5ms of CPU and then Q utilizes 5ms of CPU. Hence after 15ms P starts with I/O And after 20ms Q also starts with I/O. Since I/O can be done in parallel, P finishes I\O at 105th ms (15 + 90) and Q finishes its I\O at 110th ms (20 + 90). Therefore we can see that CPU remains idle from 20th to 105th ms.
That is when Round Robin scheduling is used,
Idle time of CPU = 85ms
CPU Utilization = 20/105 = 19.05%
When First Come First Served scheduling scheduling or Shortest Remaining Time First is used
Say P utilizes 10ms of CPU and then starts its I/O. At 11th ms Q starts processing. Q utilizes 10ms of CPU.
P completes its I/O at 100ms (10 + 90)
Q completes its I/O at 110ms (20 + 90)
At 101th ms P again utilizes CPU. Hence,
Idle time of CPU = 80ms
CPU Utilization = 20/100 = 20%

Since only two processes are involved and I\O time is much more than CPU time, “Static priority scheduling with different priorities” for the two processes reduces to FCFS or Shortest remaining time first.
Therefore, Round robin will result in least CPU utilization.

Q.No.16. Consider a set of n tasks with known runtimes r_1, r_2, …. r_n to be run on a uniprocessor machine. Which of the following processor scheduling algorithms will result in the maximum throughput?
(A) Round-Robin
(B) Shortest-Job-First
(C) Highest-Response-Ratio-Next
(D) First-Come-First-Served

Ans: (B)

Explanation:
Throughput means total number of tasks executed per unit time i.e. sum of waiting time and burst time.
Shortest job first scheduling is a scheduling policy that selects the waiting process with the smallest execution time to execute next.
Thus, in shortest job first scheduling, shortest jobs are executed first. This means CPU utilization is maximum. So, maximum number of tasks are completed.

Thus, option (B) is correct.

Q.No.17. Consider a uniprocessor system executing three tasks T1, T2 and T3, each of which is composed of an infinite sequence of jobs (or instances) which arrive periodically at intervals of 3, 7 and 20 milliseconds, respectively. The priority of each task is the inverse of its period and the available tasks are scheduled in order of priority, with the highest priority task scheduled first. Each instance of T1, T2 and T3 requires an execution time of 1, 2 and 4 milliseconds, respectively. Given that all tasks initially arrive at the beginning of the 1st milliseconds and task preemptions are allowed, the first instance of T3 completes its execution at the end of _______________ milliseconds.
(A) 5
(B) 10
(C) 12
(D) 15

Ans: (C)

Explanation: Periods of T1, T2 and T3 are 3ms, 7ms and 20ms.
Since priority is inverse of period, T1 is the highest priority task, then T2 and finally T3

Every instance of T1 requires 1ms, that of T2 requires 2ms and that of T3 requires 4ms

Initially all T1, T2 and T3 are ready to get processor, T1 is preferred

Second instances of T1, T2, and T3 shall arrive at 3, 7, and 20 respectively.

Third instance of T1, T2 and T3 shall arrive at 6, 14, and 40 respectively.

Time-Interval	Tasks
0-1	T1
1-2	T2
2-3	T2
3-4	T1 [Second Instance of T1 arrives]
4-5	T3
5-6	T3
6-7	T1 [Third Instance of T1 arrives] [Therefore T3 is preempted]
7-8	T2 [Second instance of T2 arrives]
8-9	T2
9-10	T1 [Fourth Instance of T1 arrives]
10-11	T3
11-12	T3 [**First Instance of T3 completed**]

Q.No.18. The maximum number of processes that can be in Ready state for a computer system with n CPUs is
(A) n
(B) n^2
(C) 2n
(D) Independent of n

Ans: (D)

Explanation: The size of ready queue doesn't depend on number of processes. A single processor system may have a large number of processes waiting in ready queue.

Q.No.19. For the processes listed in the following table, which of the following scheduling schemes will give the lowest average turnaround time?

Process	Arrival Time	Processing Time
A	0	3
B	1	6
C	4	4
D	6	2

(A) First Come First Serve
(B) Non-preemptive Shortest Job First
(C) Shortest Remaining Time
(D) Round Robin with Quantum value two

Ans: (C)

Explanation: Turnaround time is the total time taken between the submission of a program/process/thread/task (Linux) for execution and the return of the complete output to the customer/user.
Turnaround Time = Completion Time – Arrival Time.

FCFS = First Come First Serve (A, B, C, D)
SJF = Non-preemptive Shortest Job First (A, B, D, C)
SRT = Shortest Remaining Time (A(3), B(1), C(4), D(2), B(5))
RR = Round Robin with Quantum value 2 (A(2), B(2), A(1),C(2),B(2),D(2),C(2),B(2)

Pr	Arr.Time	P.Time	FCFS	SJF	SRT	RR
A	0	3	3-0=3	3-0=3	3-0=3	5-0=5
B	1	6	9-1=8	9-1=8	15-1=14	15-1=14

C	4	4	13-4=9	15-4=11	8-4=4	13-4=9
D	6	2	15-6=9	11-6=5	10-6=4	11-6=5
Average			7.25	6.75	6.25	8.25

Shortest Remaining Time produces minimum average turn-around time.

Q.No.20. Two concurrent processes P1 and P2 use four shared resources R1, R2, R3 and R4, as shown below.

P1	**P2**

Compute:
Use R1;
Use R2;
Use R3;
Use R4;
	Compute;
	Use R1;
	Use R2;
	Use R3;.
	Use R4;

Both processes are started at the same time, and each resource can be accessed by only one process at a time The following scheduling constraints exist between the access of resources by the processes:

- P2 must complete use of R1 before P1 gets access to R1
- P1 must complete use of R2 before P2 gets access to R2.
- P2 must complete use of R3 before P1 gets access to R3.
- P1 must complete use of R4 before P2 gets access to R4.

There are no other scheduling constraints between the processes. If only binary semaphores are used to enforce the above scheduling constraints, what is the minimum number of binary semaphores needed?

(A) 1
(B) 2
(C) 3
(D) 4

Ans: (B)

Explanation:
We use two semaphores : A and B.
A is initialized to 0 and B is initialized to 1.
P1:

```
Compute;
Wait(A);
Use R1;
Use R2;
Signal(B);
Wait(A);
Use R3;
Use R4;
Signal(B);
```

P2:

```
Compute;
Wait(B);
Use r1;
Signal(A);
Wait(B);
Use R2;
Use R3;
Signal(A);
Wait(B);
Use R4;
Signal(B);
```

In process p1, initially control will be stuck in while loop of Wait(A) because A = 0.
In process p2, Wait(B) decrements the value of B to 0 . Now, P2 uses the resource R1 and increments the value to A to 1 so that process P1 can enter its critical section and use resource R1.

Thus, P2 will complete use of R1 before P1 gets access to R1.

Now, in P2 values of B = 0. So, P2 can not use resource R2 till P1 uses R2 and calls function Signal(B) to increment the value of B to 1.
Thus, P1 will complete use of R2 before P2 gets access to R2.

Now, semaphore A = 0. So, P1 can not execute further and gets stuck in while loop of function Wait(A).
Process P2 uses R3 and increments the value of semaphore A to 1.Now, P1 can enter its critical section to use R3.
Thus, P2 will complete use of R3 before P1 gets access to R3.

Now, P1 will use R4 and increments the value of B to 1 so that P2 can enter is critical section to use R4.
Thus, P1 will complete use of R4 before P2 gets access to R4.

Thus, option (B) is correct.

Q.No.21. We wish to schedule three processes P1, P2 and P3 on a uniprocessor system. The priorities, CPU time requirements and arrival times of the processes are as shown below.

Process	Priority	CPU time required	Arrival time (hh:mm:ss)
P1	10(highest)	20 sec	00:00:05
P2	9	10 sec	00:00:03
P3	8 (lowest)	15 sec	00:00:00

We have a choice of preemptive or non-preemptive scheduling. In preemptive scheduling, a late-arriving higher priority process can preempt a currently running process with lower priority. In non-preemptive scheduling, a late-arriving higher priority process must wait for the currently executing process to complete before it can be scheduled on the processor.

What are the turnaround times (time from arrival till completion) of P2 using preemptive and non-preemptive scheduling respectively.

(A)30 sec, 30 sec
(B)30 sec, 10 sec
(C)42 sec, 42 sec
(D)30 sec, 42 sec

Ans: (D)

Explanation:
For Non preemptive scheduling

P3(AT=0)	P1(AT=5)	P2(**AT=3**)

0 15 35 **45**
Turn Around Time= Completion Time – Arrival Time = 45 -3 = 42

For Preemptive scheduling

P3	P3	P3	P2	P2	P1	P2	P3

0 1 2 **3** 4 5 25 **33** 45
Turn Around Time= Completion Time – Arrival Time = 33 – 3 = 30

Q.No.22. Consider an arbitrary set of CPU-bound processes with unequal CPU burst lengths submitted at the same time to a computer system. Which one of the following process scheduling algorithms would minimize the average waiting time in the ready queue?

(A) Shortest remaining time first
(B) Round-robin with time quantum less than the shortest CPU burst
(C) Uniform random
(D) Highest priority first with priority proportional to CPU burst length

Ans: (A)

Explanation:
Turnaround time is the total time taken by the process between starting and the completion and waiting time is the time for which process is ready to run but not executed by CPU scheduler. As we know, in all CPU Scheduling algorithms, shortest job first is optimal i.ie. it gives minimum turn round time, minimum average waiting time and high throughput and the most important thing is that shortest remaining time first is the pre-emptive version of shortest job first.

shortest remaining time first scheduling algorithm may lead to starvation because If the short processes are added to the cpu scheduler continuously then the currently running process will never be able to execute as they will get pre-empted but here all the processes are arrived at same time so there will be no issue such as starvation. So, the answer is Shortest remaining time first, which is answer (A).

Q.No.23. Consider the following processes, with the arrival time and the length of the CPU burst given in milliseconds. The scheduling algorithm used is preemptive shortest remaining-time first.

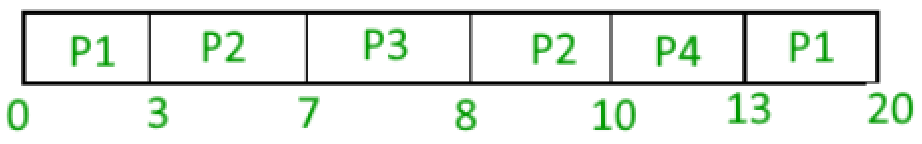
Process	Arrival Time	Burst Time
P_1	0	10
P_2	3	6
P_3	7	1
P_4	8	3

The average turnaround time of these processes is ___________ milliseconds.

(A) 8.25
(B) 10.25
(C) 6.35
(D) 4.25

Ans: (A)

Explanation: Preemptive Shortest Remaining time first scheduling, i.e. that processes will be scheduled on the CPU which will be having least remaining burst time(required time at the CPU).
The processes are scheduled and executed as given in the below Gantt chart.



Gantt chart for the process getting scheduled and executed at the CPU

Turn Around Time(TAT) = Completion Time(CT) – Arrival Time(AT)

TAT for P1 = 20 – 0 = 20

TAT for P2 = 10 – 3 = 7

TAT for P3 = 8- 7 = 1

TAT for P4 = 13 – 8 = 5

Hence, Average TAT = Total TAT of all the processes / no of processes = (20 + 7 + 1 + 5) / 4 = 33 / 4 = 8.25

Thus, option (A) is the correct choice.

Q.No.24. Assume every process requires 3 seconds of service time in a system with single processor. If new processes are arriving at the rate of 10 processes per minute, then estimate the fraction of time CPU is busy in system?
(A) 20%
(B) 30%
(C) 50%
(D) 60%

Ans: (C)

Explanation:
10 processes -> 1 min
1 process-> 1/10 min = 6 sec (Arrival rate)

Each process -> 3 sec service time 3/6 * 100 = 50% of time CPU is busy.

Q.No.25. Consider *n* processes sharing the CPU in a round-robin fashion. Assuming that each process switch takes *s* seconds, what must be the quantum size *q* such that the overhead resulting from process switching is minimized but, at the same time, each process is guaranteed to get its turn at the CPU at least every *t* seconds?

A. $q \le \frac{t-ns}{n-1}$

B. $q \ge \frac{t-ns}{n-1}$

C. $q \le \frac{t-ns}{n+1}$

D. $q \ge \frac{t-ns}{n+1}$

Ans: (A)

Explanation: Each process will get CPU for q seconds and each process wants CPU again after t seconds.
Thus, there will be (n-1) processes once after current process gets CPU again.
Each process takes s seconds for context switch.
(Qp1)(s)(Qp2)(s)(Qp3)(s)(Qp1)

It can be seen that since P1 left and arrived again, there have been n context switches and (n-1) processes. Thus, equation will be:

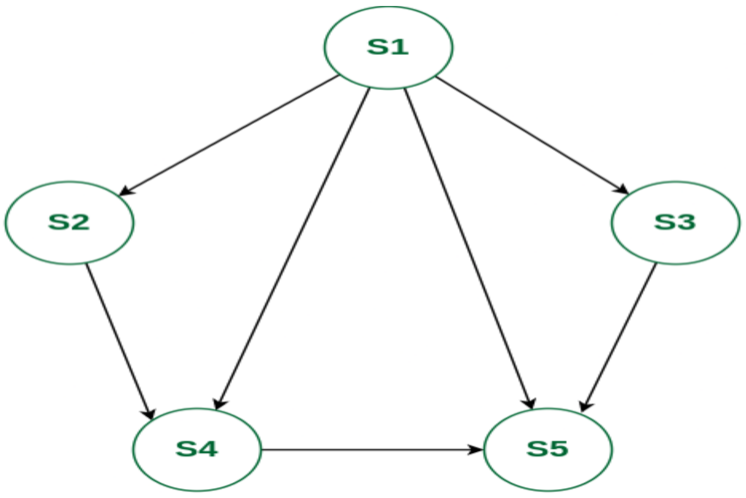
q*(n-1) + n*s <= t

q*(n-1) <= t - n*s

q <= (t-n.s) / (n-1)

So, option (A) is correct.

Q.No.26. Answer the following: a. Four jobs are waiting to be run. Their expected run times are 6, 3, 5 and x. In what order should they be run to minimize the average response time? b. Write a concurrent program using ***par begin – par end*** to represent the precedence graph shown below.



Ans :

Explanation:
a. If we use Round Robin with Time Quantum = 1, it will take minimum Response Time, but we will not get any specific order. So, **SJF** (Shortest Job First) will give a minimum response time.

- if x<3<5<6 then order: x, 3, 5, 6
- if 3<x<5<6 then order: 3, x, 5, 6
- if 3<5<x<6 then order: 3, 5, x, 6
- if 3<5<6<x then order: 3, 5, 6, x

b.

```
Begin
    S1;
    Par Begin
         S3;
         Begin
              S2;
              S3;
         End
    Par End
    S5;
End
```

Q.No.27. When an interrupt occurs, an operating system
(A) ignores the interrupt
(B) always changes state of interrupted process to ‘blocked’ and schedules another process
(C) always resumes execution of interrupted process after processing the interrupt
(D) may change the state of interrupted process to ‘blocked’and schedule another process

Ans: (D)

Explanation: Scheduler decides that the interrupted process will complete execution or some other process will be executed.
If the interrupt signaled an I/O completion event, and at the same time a high priority process came into Ready state then the scheduler block the interrupted process and dispatch the high priority process in the running state. If low priority process comes into Ready state then scheduler dispatch the interrupted process. Hence, **D is correct**.

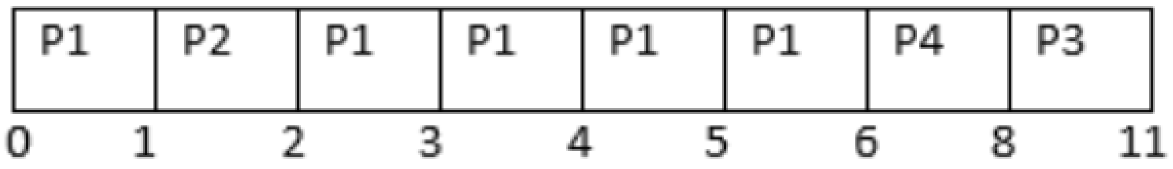
Q.No.28. Consider the following CPU processes with arrival times (in milliseconds) and length of CPU bursts (in milliseconds) except for process P4 as given below:

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

If the average waiting time across all processes is 2 milliseconds and pre-emptive shortest remaining time first scheduling algorithm is used to schedule the processes, then find the value of x ?
(A) 1
(B) 2
(C) 4
(D) None of these

Ans: (B)

Explanation: If we take value of x is 2, then we have gantt chart as



So, completion time of P1, P2, P3, and P4 are 6, 2, 11, and 8 respectively.
Turn around time of P1, P2, P3, and P4 are 6, 1, 8, and 4 respectively.
Waiting time of P1, P2, P3, and P4 are 1, 0, 5, and 2 resectively.
Therefore, average waiting time = (1+0+5+2) / 4 = 8/2 = 2

Option (B) is correct.

Q.No.29. Consider the following four processes with the arrival time and length of CPU burst given in milliseconds :

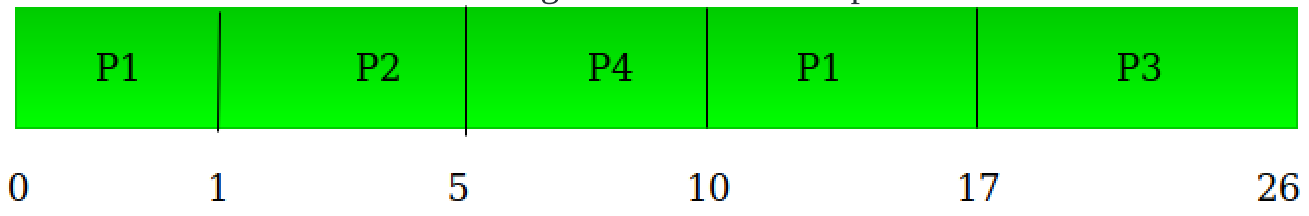
Process	Arrival Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

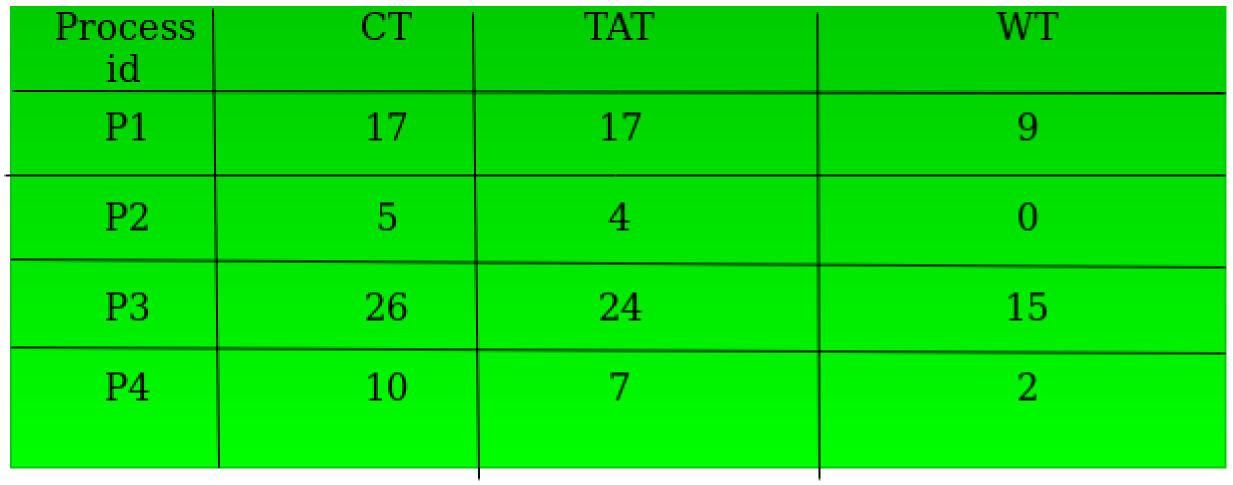
The average waiting time for preemptive SJF scheduling algorithm is ___________.

(A) 6.5
(B) 7.5
(C) 6.75
(D) 7.75

Ans: (A)

Explanation: First we will make gantt chart of given process then we will calculate turn around time and waiting time of individual process.





Process id	CT	TAT	WT
P1	17	17	9
P2	5	4	0
P3	26	24	15
P4	10	7	2

Now we have to calculate average waiting time for schedule:
avg waiting time = wt(P1 + P2 + P3 + P4)/number of process.

ie. (9 + 0 + 15 + 2) / 4

= 26 / 4

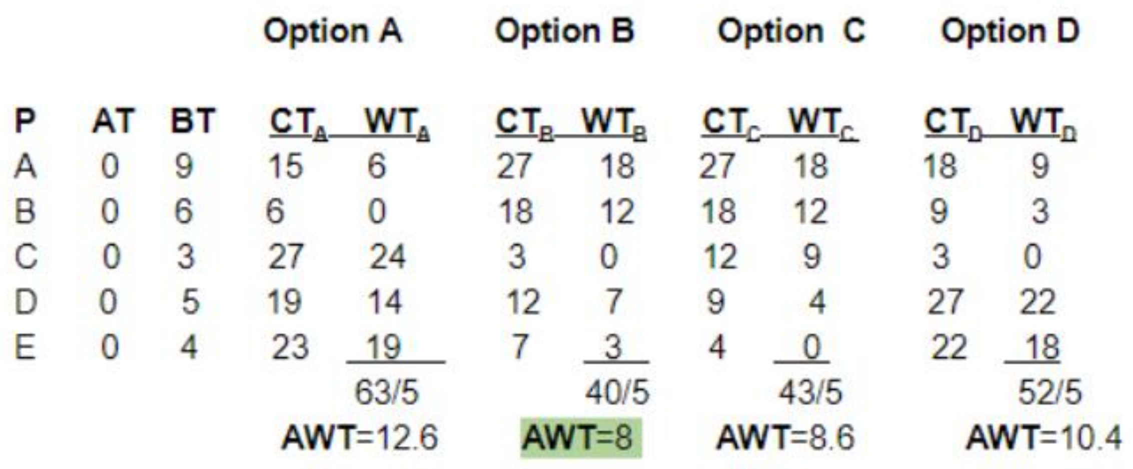
= 6.5

So, option (A) is correct.

Q.No.30. Five jobs A, B, C, D and E are waiting in Ready Queue. Their expected runtimes are 9, 6, 3, 5 and x respectively. All jobs entered in Ready queue at time zero. They must run in __ order to minimize average response time if $3 < x < 5$.
(A) B, A, D, E, C
(B) C, E, D, B, A
(C) E, D, C, B, A
(D) C, B, A, E, D

Ans: (B)

Explanation: We will solve it by minimizing the avg Waiting Time and Take x = 4
BT = Burst or Execution time
CT = completion time
P = Process
AWT = Average Waiting Time
AT = Arrival time
WT = Waiting time



			Option A		Option B		Option C		Option D	
P	AT	BT	CT_A	WT_A	CT_B	WT_B	CT_C	WT_C	CT_D	WT_D
A	0	9	15	6	27	18	27	18	18	9
B	0	6	6	0	18	12	18	12	9	3
C	0	3	27	24	3	0	12	9	3	0
D	0	5	19	14	12	7	9	4	27	22
E	0	4	23	19	7	3	4	0	22	18
				63/5		40/5		43/5		52/5
			AWT=12.6		AWT=8		AWT=8.6		AWT=10.4	

We get minimum average waiting time from the sequence given in option (B).

Q.No.31. A system has four processes and five allocatable resources. The current allocation and maximum needs are as follows:

	Allocated	Maximum	Available
Process A	1 0 2 1 1	1 1 2 1 3	0 0 x 1 1
Process B	2 0 1 1 0	2 2 2 1 0	
Process C	1 1 0 1 0	2 1 3 1 0	
Process D	1 1 1 1 0	1 1 2 2 1	

The smallest value of x for which the above system in safe state is ___________.

(A) 1
(B) 3
(C) 2
(D) Not safe for any value of x.

Ans: (D)

Explanation:

	Allocated	Max	Available	Need
Process A	1 0 2 1 1	1 1 2 1 3	0 0 x 1 1	0 1 0 0 2
Process B	2 0 1 1 0	2 2 2 1 0		0 2 1 0 0
Process C	1 1 0 1 0	2 1 3 1 0		1 0 3 0 0
Process D	1 1 1 1 0	1 1 2 2 1		0 0 1 1 1

The smallest value of x for which the above system in safe state is ___________.

For x = 1 process D will execute and free 1 1 2 2 1 instances. Now none of the other process will execute.
let x = 2 then process D will execute and free 1 1 3 2 1 instances. Now process C will execute and free 2 2 3 3 1 instances. With these free instances process B will execute, but process A will not execute because 5 resources needs 2 instances which will never be satisfied. That's why system is not in safe state.
So, option (D) is correct.

Q.No.32. In a system using single processor, a new process arrives at the rate of six processes per minute and each such process requires seven seconds of service time. What is the CPU utilization?
(A) 70%
(B) 30%
(C) 60%
(D) 64%

Ans: (A)

Explanation: Number of processes per minute = 6
Burst time of each process = 7 secs

CPU utilization time within a minute = 6*7 = 42 secs

% CPU utilization = useful time / total time * 100

= (42/60) * 100

= 70%

Option (A) is correct.

Q.No.33. Which of the following statements is not true for Multi Level Feedback Queue processor scheduling algorithm?
(A) Queues have different priorities
(B) Each queue may have different scheduling algorithm
(C) Processes are permanently assigned to a queue
(D) This algorithm can be configured to match a specific system under design

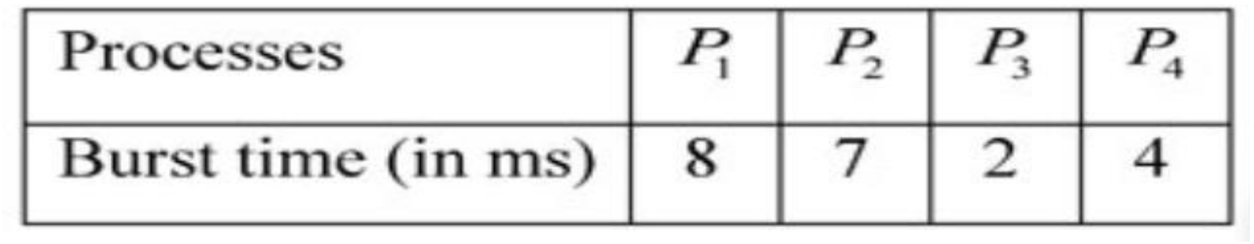
Ans: (C)

Explanation: For Multi Level Feedback Queue processor scheduling algorithm:

- Queues have different priorities
- Each queue may have different scheduling algorithm
- Processes are not permanently assigned to a queue.
- This algorithm can be configured to match a specific system under design

Option (C) is correct.

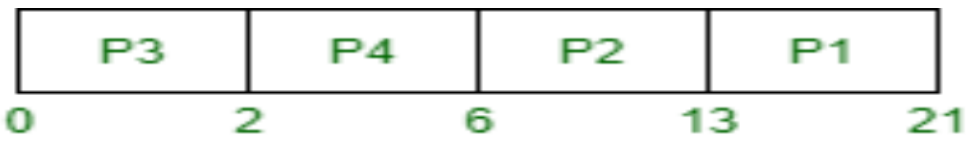
Q.No.34. Consider the following set of processes, assumed to have arrived at time 0. Consider the CPU scheduling algorithms Shortest Job First (SJF) and Round Robin (RR). For RR, assume that the processes are scheduled in the order P_1, P_2, P_3, P_4.



Processes	P_1	P_2	P_3	P_4
Burst time (in ms)	8	7	2	4

If the time quantum for RR is 4 ms, then the absolute value of the difference between the average turnaround times (in ms) of SJF and RR (round off to 2 decimal places) is __________ .
(A) 5.0
(B) 5.25
(C) 5.50
(D) 5.75

Ans: (B)



Therefore, Average Turn Around Time (TAT) is,

= {(21 – 0) + (13 – 0) + (2 – 0) + (6 – 0)} / 4

= 10.5

Now, according to Round Robin (RR) CPU Scheduling with time quantum 4, gantt chart is,

Therefore, Average Turn Around Time (TAT) is,

= {(18 – 0) + (21 – 0) + (10 – 0) + (14 – 0)} / 4

= 15.75

Hence,

= □SJF (TAT) – RR(TAT)□

= □10.5 – 15.75□

= 5.25

Option (B) is correct.

<u>Deadlock Detection-</u>

Using Resource Allocation Graph, it can be easily detected whether system is in a **Deadlock** state or not.

The rules are-

Rule-01:

In a Resource Allocation Graph where all the resources are single instance,

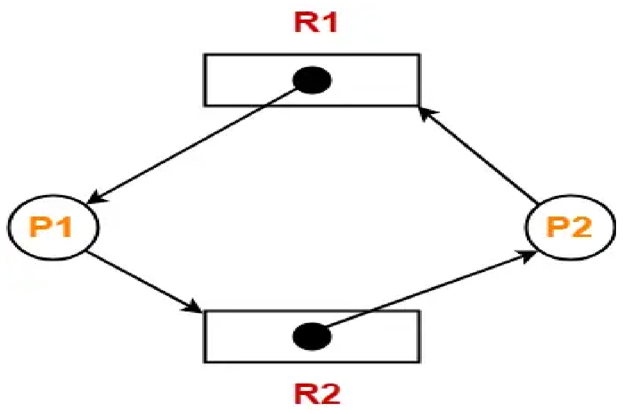
- If a cycle is being formed, then system is in a deadlock state.
- If no cycle is being formed, then system is not in a deadlock state.

Rule-02:

In a Resource Allocation Graph where all the resources are **NOT** single instance,

- If a cycle is being formed, then system may be in a deadlock state.
- **Banker's Algorithm** is applied to confirm whether system is in a deadlock state or not.
- If no cycle is being formed, then system is not in a deadlock state.
- Presence of a cycle is a necessary but not a sufficient condition for the occurrence of deadlock.

Q.No.35. Consider the resource allocation graph in the figure-



Ans:

<u>Method-01:</u>

- The given resource allocation graph is single instance with a cycle contained in it.
- Thus, the system is definitely in a deadlock state.

<u>Method-02:</u>

Using the given resource allocation graph, we have-

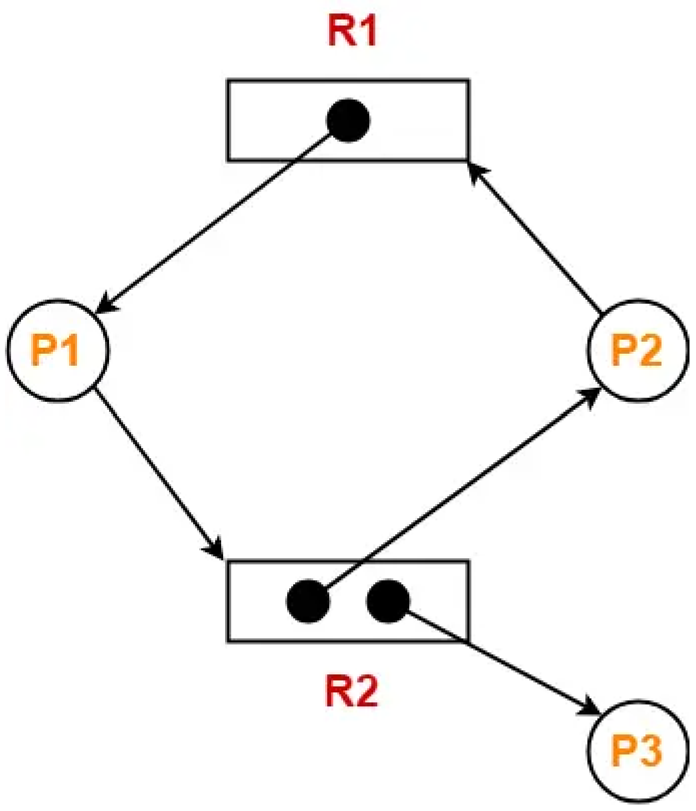
	Allocation		**Need**	
	R1	**R2**	**R1**	**R2**
Process P1	1	0	0	1
Process P2	0	1	1	0

Available = [R1 R2] = [0 0]

Now,

- There are no instances available currently and both the processes require a resource to execute.
- Therefore, none of the process can be executed and both keeps waiting infinitely.
- Thus, the system is in a deadlock state.

Q.No.36. Consider the resource allocation graph in the figure-



Find if the system is in a deadlock state otherwise find a safe sequence.

<u>Ans:</u>

- The given resource allocation graph is multi instance with a cycle contained in it.
- So, the system may or may not be in a deadlock state.

Using the given resource allocation graph, we have-

	Allocation		**Need**	
	R1	**R2**	**R1**	**R2**
Process P1	1	0	0	1
Process P2	0	1	1	0
Process P3	0	1	0	0

Available = [R1 R2] = [0 0]

<u>Step-01:</u>

- Since process P3 does not need any resource, so it executes.
- After execution, process P3 release its resources.

Then,

Available

= [0 0] + [0 1]

= [0 1]

<u>Step-02:</u>

- With the instances available currently, only the requirement of the process P1 can be satisfied.
- So, process P1 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [0 1] + [1 0]

= [1 1]

Step-03:

- With the instances available currently, the requirement of the process P2 can be satisfied.
- So, process P2 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

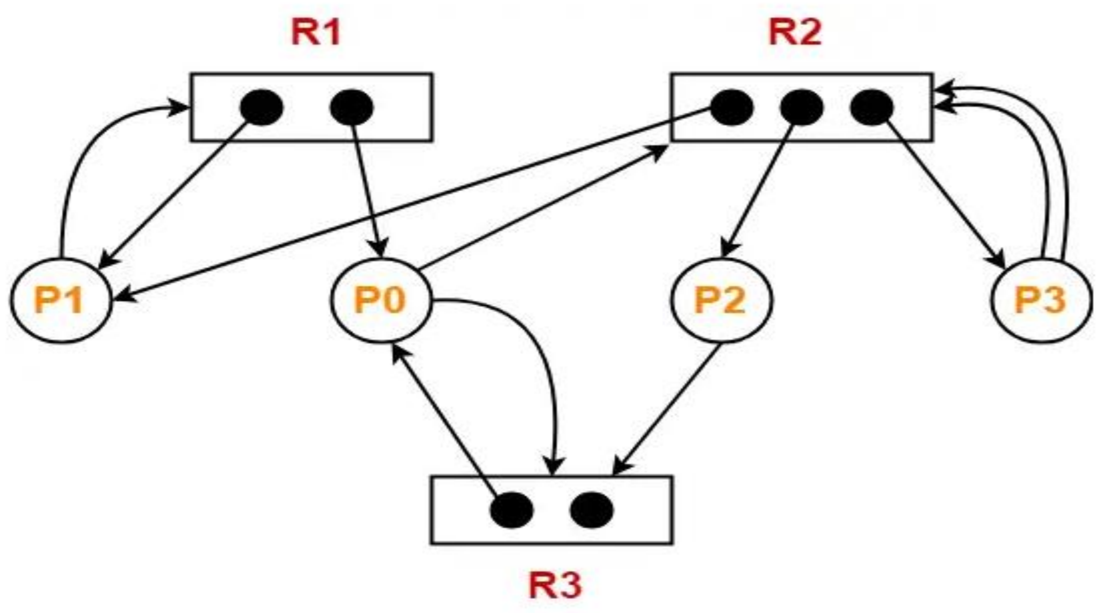
= [1 1] + [0 1]

= [1 2]

Thus,

- There exists a safe sequence P3, P1, P2 in which all the processes can be executed.
- So, the system is in a safe state.

Q.No.37. Consider the resource allocation graph in the figure-



Find if the system is in a deadlock state otherwise find a safe sequence.

Ans:

- The given resource allocation graph is multi instance with a cycle contained in it.
- So, the system may or may not be in a deadlock state.

Using the given resource allocation graph, we have-

	Allocation			Need		
	R1	**R2**	**R3**	**R1**	**R2**	**R3**
Process P0	1	0	1	0	1	1
Process P1	1	1	0	1	0	0
Process P2	0	1	0	0	0	1
Process P3	0	1	0	0	2	0

Available = [R1 R2 R3] = [0 0 1]

Step-01:

- With the instances available currently, only the requirement of the process P2 can be satisfied.
- So, process P2 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [0 0 1] + [0 1 0]

= [0 1 1]

Step-02:

- With the instances available currently, only the requirement of the process P0 can be satisfied.
- So, process P0 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [0 1 1] + [1 0 1]

= [1 1 2]

Step-03:

- With the instances available currently, only the requirement of the process P1 can be satisfied.
- So, process P1 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [1 1 2] + [1 1 0]

= [2 2 2]

Step-04:

- With the instances available currently, the requirement of the process P3 can be satisfied.
- So, process P3 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

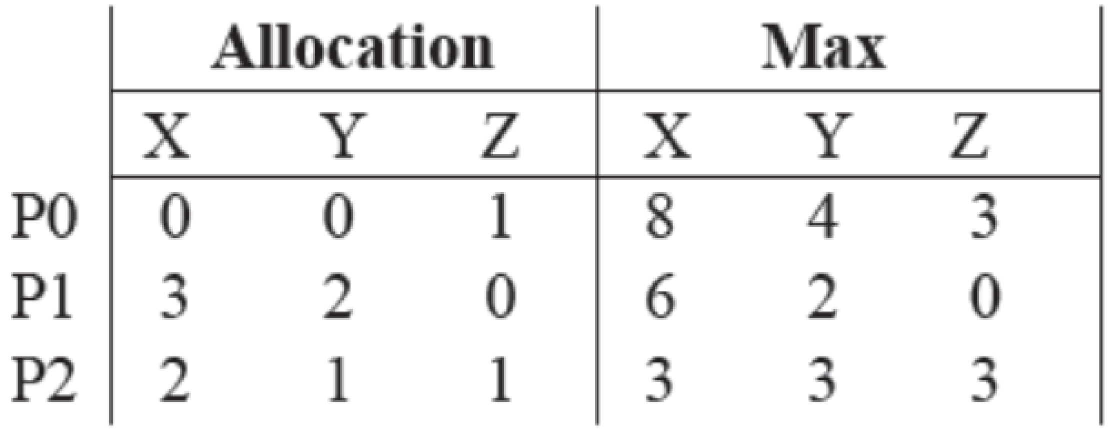
= [2 2 2] + [0 1 0]

= [2 3 2]

Thus,

- There exists a safe sequence P2, P0, P1, P3 in which all the processes can be executed.
- So, the system is in a safe state.

Q.No.38. An operating system uses the Banker's algorithm for deadlock avoidance when managing the allocation of three resource types X, Y, and Z to three processes P0, P1, and P2. The table given below presents the current system state. Here, the Allocation matrix shows the current number of resources of each type allocated to each process and the Max matrix shows the maximum number of resources of each type required by each process during its execution.



	Allocation			Max		
	X	Y	Z	X	Y	Z
P0	0	0	1	8	4	3
P1	3	2	0	6	2	0
P2	2	1	1	3	3	3

There are 3 units of type X, 2 units of type Y and 2 units of type Z still available. The system is currently in a safe state. Consider the following independent requests for additional resources in the
current state:

REQ1: P0 requests 0 units of X,

0 units of Y and 2 units of Z

REQ2: P1 requests 2 units of X,

0 units of Y and 0 units of Z

Which one of the following is TRUE?
(A) Only REQ1 can be permitted.
(B) Only REQ2 can be permitted.
(C) Both REQ1 and REQ2 can be permitted.
(D) Neither REQ1 nor REQ2 can be permitted

Ans: (B)

Explanation: This is the current safe state.

AVAILABLE	X=3, Y=2, Z=2	
	MAX	ALLOCATION
	X Y Z	X Y Z
P0	8 4 3	0 0 1
P1	6 2 0	3 2 0
P2	3 3 3	2 1 1

Now, if the request REQ1 is permitted, the state would become :

	AVAILABLE	X=3, Y=2, Z=0	
	MAX	ALLOCATION	NEED
	X Y Z	X Y Z	X Y Z
P0	8 4 3	0 0 3	8 4 0
P1	6 2 0	3 2 0	3 0 0
P2	3 3 3	2 1 1	1 2 2

Now, with the current availability, we can service the need of P1. The state would become :

	AVAILABLE	X=6, Y=4, Z=0	
	MAX	ALLOCATION	NEED
	X Y Z	X Y Z	X Y Z
P0	8 4 3	0 0 3	8 4 0
P1	6 2 0	3 2 0	0 0 0
P2	3 3 3	2 1 1	1 2 2

With the resulting availability, it would not be possible to service the need of either P0 or P2, owing to lack of Z resource.

Therefore, the system would be in a deadlock.

⇒ We cannot permit REQ1.

Now, at the given safe state, if we accept REQ2 :

	AVAILABLE	X=1, Y=2, Z=2	
	MAX	ALLOCATION	NEED
	X Y Z	X Y Z	X Y Z
P0	8 4 3	0 0 1	8 4 2
P1	6 2 0	5 2 0	1 0 0
P2	3 3 3	2 1 1	1 2 2

With this availability, we service P1 (P2 can also be serviced). So, the state is :

	AVAILABLE	X=6, Y=4, Z=2	
	MAX	ALLOCATION	NEED
	X Y Z	X Y Z	X Y Z
P0	8 4 3	0 0 1	8 4 2

P1	6 2 0	5 2 0	0 0 0
P2	3 3 3	2 1 1	1 2 2

With the current availability, we service P2. The state becomes :

	AVAILABLE	X=8, Y=5, Z=3	
	MAX	ALLOCATION	NEED
	X Y Z	X Y Z	X Y Z
P0	8 4 3	0 0 1	8 4 2
P1	6 2 0	5 2 0	0 0 0
P2	3 3 3	2 1 1	0 0 0

Finally, we service P0. The state now becomes :

	AVAILABLE	X=8, Y=5, Z=4	
	MAX	ALLOCATION	NEED
	X Y Z	X Y Z	X Y Z
P0	8 4 3	0 0 1	0 0 0

P1	6 2 0	5 2 0	0 0 0
P2	3 3 3	2 1 1	0 0 0

The state so obtained is a safe state. ⇒ REQ2 can be permitted.

So, only REQ2 can be permitted.

Hence, B is the correct choice.

Q.No.39. A single processor system has three resource types X, Y and Z, which are shared by three processes. There are 5 units of each resource type. Consider the following scenario, where the column alloc denotes the number of units of each resource type allocated to each process, and the column request denotes the number of units of each resource type requested by a process in order to complete execution. Which of these processes will finish LAST?

1. P0
2. P1
3. P2
4. None of the above since the system is in a deadlock

	Alloc			**Request**		
	X	**Y**	**Z**	**X**	**Y**	**Z**
P0	1	2	1	1	0	3
P1	2	0	1	0	1	2
P2	2	2	1	1	2	0

Ans:

According to question-

- Total = [X Y Z] = [5 5 5]
- Total _Alloc = [X Y Z] = [5 4 3]

Now,

Available

= Total – Total_Alloc

= [5 5 5] – [5 4 3]

= [0 1 2]

Step-01:

- With the instances available currently, only the requirement of the process P1 can be satisfied.
- So, process P1 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then,

Available

= [0 1 2] + [2 0 1]

= [2 1 3]

Step-02:

- With the instances available currently, only the requirement of the process P0 can be satisfied.
- So, process P0 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [2 1 3] + [1 2 1]

= [3 3 4]

Step-03:

- With the instances available currently, the requirement of the process P2 can be satisfied.
- So, process P2 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [3 3 4] + [2 2 1]

= [5 5 5]

Thus,

- There exists a safe sequence P1, P0, P2 in which all the processes can be executed.
- So, the system is in a safe state.
- Process P2 will be executed at last.

Thus, Option (3) is correct.

Q.No.40. An operating system uses the banker's algorithm for deadlock avoidance when managing the allocation of three resource types X, Y and Z to three processes P0, P1 and P2. The table given below presents the current system state. Here, the Allocation matrix shows the current number of resources of each type allocated to each process and the Max matrix shows the maximum number of resources of each type required by each process during its execution.

	Allocation			**Max**		
	X	**Y**	**Z**	**X**	**Y**	**Z**
P0	0	0	1	8	4	3
P1	3	2	0	6	2	0
P2	2	1	1	3	3	3

There are 3 units of type X, 2 units of type Y and 2 units of type Z still available. The system is currently in safe state. Consider the following independent requests for additional resources in the current state-

REQ1: P0 requests 0 units of X, 0 units of Y and 2 units of Z

REQ2: P1 requests 2 units of X, 0 units of Y and 0 units of Z

Which of the following is TRUE?

1. Only REQ1 can be permitted
2. Only REQ2 can be permitted
3. Both REQ1 and REQ2 can be permitted
4. Neither REQ1 nor REQ2 can be permitted

Ans :

According to question,

Available = [X Y Z] = [3 2 2]

Now,

Need = Max – Allocation

So, we have-

	Allocation			**Max**			**Need**		
	X	**Y**	**Z**	**X**	**Y**	**Z**	**X**	**Y**	**Z**
P0	0	0	1	8	4	3	8	4	2
P1	3	2	0	6	2	0	3	0	0
P2	2	1	1	3	3	3	1	2	2

Currently, the system is in safe state.

Checking Whether REQ1 Can Be Entertained-

- Need of P0 = [0 0 2]
- Available = [3 2 2]

Clearly,

- With the instances available currently, the requirement of REQ1 can be satisfied.
- So, banker’s algorithm assumes that the request REQ1 is entertained.
- It then modifies its data structures as-

	Allocation			**Max**			**Need**		
	X	**Y**	**Z**	**X**	**Y**	**Z**	**X**	**Y**	**Z**
P0	0	0	**3**	8	4	3	8	4	**0**
P1	3	2	0	6	2	0	3	0	0
P2	2	1	1	3	3	3	1	2	2

Available

= [3 2 2] – [0 0 2]

= [3 2 0]

- Now, it follows the safety algorithm to check whether this resulting state is a safe state or not.
- If it is a safe state, then REQ1 can be permitted otherwise not.

<u>Step-01:</u>

- With the instances available currently, only the requirement of the process P1 can be satisfied.
- So, process P1 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.
  - Then-
  - Available
  - = [3 2 0] + [3 2 0]
  - = [6 4 0]

Now,

- It is not possible to entertain any process.
- The system has entered the deadlock state which is an unsafe state.
- Thus, REQ1 will not be permitted.

Checking Whether REQ2 Can Be Entertained-

- Need of P1 = [2 0 0]
- Available = [3 2 2]

Clearly,

- With the instances available currently, the requirement of REQ1 can be satisfied.
- So, banker's algorithm assumes the request REQ2 is entertained.
- It then modifies its data structures as-

	Allocation			**Max**			**Need**		
	X	**Y**	**Z**	**X**	**Y**	**Z**	**X**	**Y**	**Z**
P0	0	0	1	8	4	3	8	4	2
P1	**5**	2	0	6	2	0	**1**	0	0
P2	2	1	1	3	3	3	1	2	2

Available

= [3 2 2] – [2 0 0]

- = [1 2 2]
  Now, it follows the safety algorithm to check whether this resulting state is a safe state or not.
- If it is a safe state, then REQ2 can be permitted otherwise not.

Step-01:

- With the instances available currently, only the requirement of the process P1 can be satisfied.
- So, process P1 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [1 2 2] + [5 2 0]

= [6 4 2]

Step-02:

- With the instances available currently, only the requirement of the process P2 can be satisfied.
- So, process P2 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [6 4 2] + [2 1 1]

= [8 5 3]

Step-03:

- With the instances available currently, the requirement of the process P0 can be satisfied.
- So, process P0 is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [8 5 3] + [0 0 1]

= [8 5 4]

Thus,

- There exists a safe sequence P1, P2, P0 in which all the processes can be executed.
- So, the system is in a safe state.
- Thus, REQ2 can be permitted.

Thus, Correct Option is (2).

Q.No.41. A system has 4 processes and 5 allocatable resource. The current allocation and maximum needs are as follows-

	Allocated					**Maximum**				
A	1	0	2	1	1	1	1	2	1	3
B	2	0	1	1	0	2	2	2	1	0
C	1	1	0	1	1	2	1	3	1	1
D	1	1	1	1	0	1	1	2	2	0

If Available = [0 0 X 1 1], what is the smallest value of x for which this is a safe state?

Ans:

Let us calculate the additional instances of each resource type needed by each process.

We know,

Need = Maximum – Allocation

So, we have-

	Need				
A	0	1	0	0	2
B	0	2	1	0	0

C	1	0	3	0	0
D	0	0	1	1	0

Case-01: For X = 0

If X = 0, then-

Available

= [0 0 0 1 1]

- With the instances available currently, the requirement of any process can not be satisfied.
- So, for X = 0, system remains in a deadlock which is an unsafe state.

Case-02: For X = 1

If X = 1, then-

Available

= [0 0 1 1 1]

Step-01:

- With the instances available currently, only the requirement of the process D can be satisfied.
- So, process D is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [0 0 1 1 1] + [1 1 1 1 0]

= [1 1 2 2 1]

- With the instances available currently, the requirement of any process can not be satisfied.

Case-02: For X = 2

If X = 2, then-

Available

= [0 0 2 1 1]

Step-01:

- With the instances available currently, only the requirement of the process D can be satisfied.
- So, process D is allocated the requested resources.
- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [0 0 2 1 1] + [1 1 1 1 0]

= [1 1 3 2 1]

Step-02:

- With the instances available currently, only the requirement of the process C can be satisfied.
- So, process C is allocated the requested resources.

- It completes its execution and then free up the instances of resources held by it.

Then-

Available

= [1 1 3 2 1] + [1 1 0 1 1]

= [2 2 3 3 2]

Step-03:

- With the instances available currently, the requirement of both the processes A and B can be satisfied.
- So, processes A and B are allocated the requested resources one by one.
- They complete their execution and then free up the instances of resources held by it.

Then-

Available

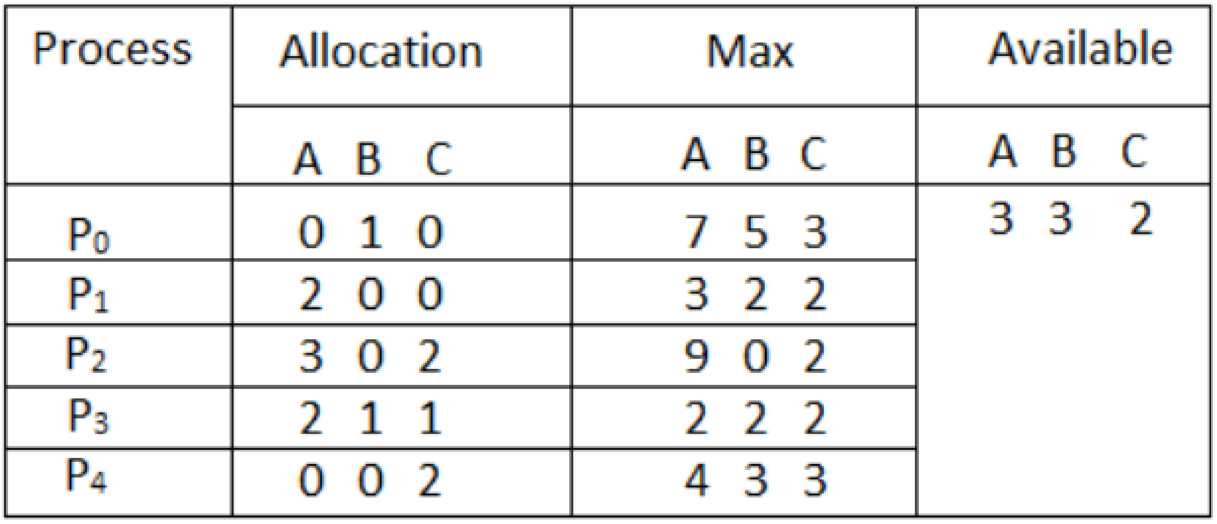
= [2 2 3 3 2] + [1 0 2 1 1] + [2 0 1 1 0]

= [5 2 6 5 3]

Thus,

- There exists a safe sequence in which all the processes can be executed.
- So, the system is in a safe state.
- Thus, minimum value of X that ensures system is in safe state = 2.

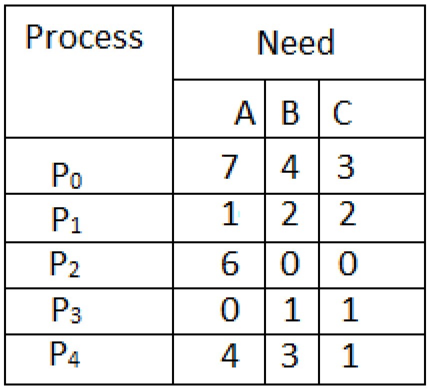
Q.No.42. Considering a system with five processes P_0 through P_4 and three resources of type A, B, C. Resource type A has 10 instances, B has 5 instances and type C has 7 instances. Suppose at time t_0 following snapshot of the system has been taken:



Process	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	3	3	2
P_1	2	0	0	3	2	2			
P_2	3	0	2	9	0	2			
P_3	2	1	1	2	2	2			
P_4	0	0	2	4	3	3			

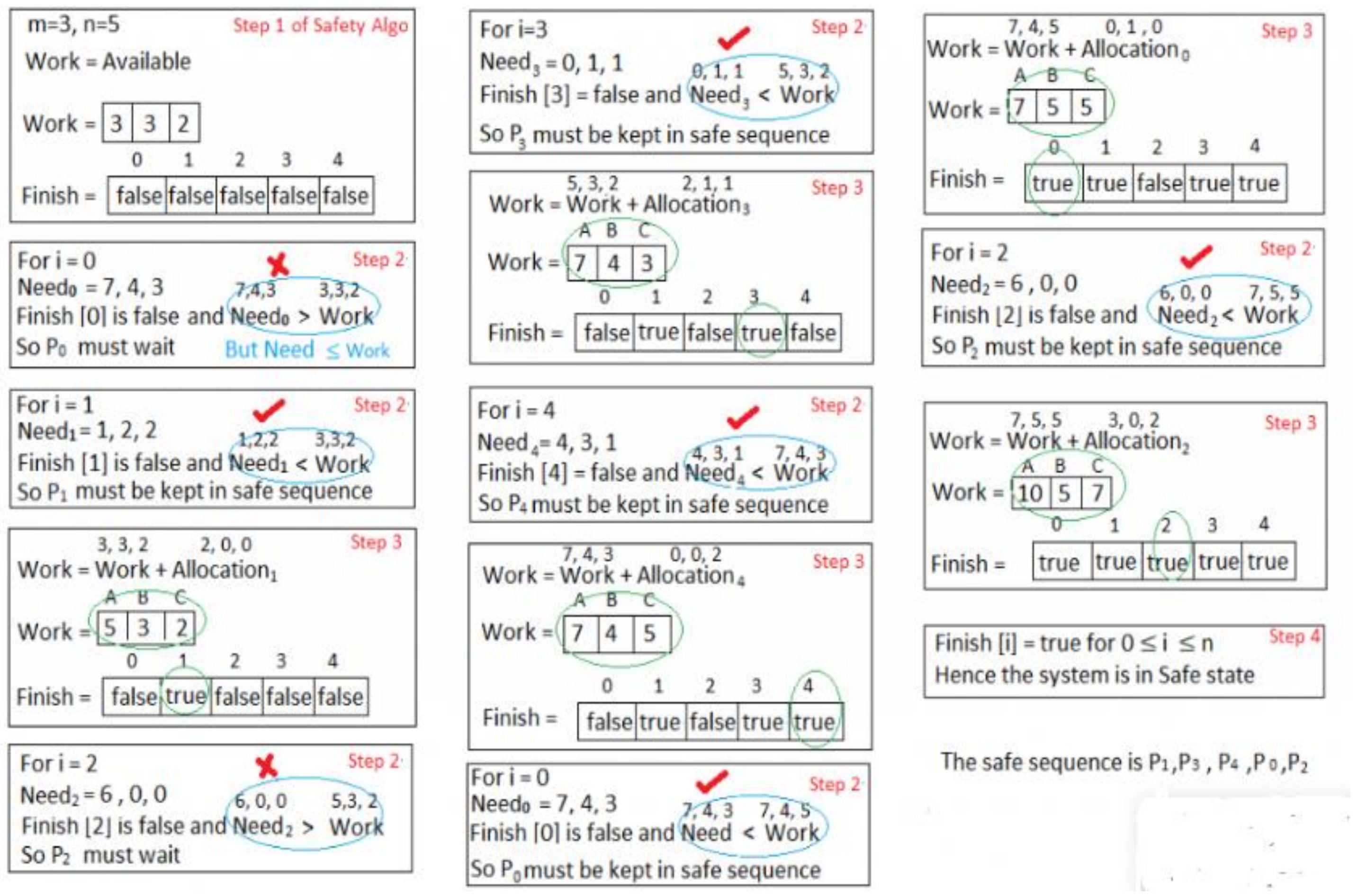
(A) What will be the content of the Need matrix?

Need [i, j] = Max [i, j] – Allocation [i, j]
So, the content of Need Matrix is:



Process	Need		
	A	B	C
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

(B) Is the system in a safe state? If Yes, then what is the safe sequence?
Applying the Safety algorithm on the given system,



m=3, n=5 Step 1 of Safety Algo

Work = Available

Work =

3	3	2

Finish =

0	1	2	3	4
false	false	false	false	false

For i = 0 Step 2 ✗

$Need_0$ = 7, 4, 3

Finish [0] is false and $\overset{7,4,3}{Need_0} > \overset{3,3,2}{Work}$

So P_0 must wait But Need $\leq$ Work

For i = 1 Step 2 ✓

$Need_1$ = 1, 2, 2

Finish [1] is false and $\overset{1,2,2}{Need_1} < \overset{3,3,2}{Work}$

So P_1 must be kept in safe sequence

Step 3

Work = $\overset{3,3,2}{Work} + \overset{2,0,0}{Allocation_1}$

Work =

A	B	C
5	3	2

Finish =

0	1	2	3	4
false	true	false	false	false

For i = 2 Step 2 ✗

$Need_2$ = 6 , 0, 0

Finish [2] is false and $\overset{6,0,0}{Need_2} > \overset{5,3,2}{Work}$

So P_2 must wait

For i=3 Step 2 ✓

$Need_3$ = 0, 1, 1

Finish [3] = false and $\overset{0,1,1}{Need_3} < \overset{5,3,2}{Work}$

So P_3 must be kept in safe sequence

Step 3

Work = $\overset{5,3,2}{Work} + \overset{2,1,1}{Allocation_3}$

Work =

A	B	C
7	4	3

Finish =

0	1	2	3	4
false	true	false	true	false

For i = 4 Step 2 ✓

$Need_4$= 4, 3, 1

Finish [4] = false and $\overset{4,3,1}{Need_4} < \overset{7,4,3}{Work}$

So P_4 must be kept in safe sequence

Step 3

Work = $\overset{7,4,3}{Work} + \overset{0,0,2}{Allocation_4}$

Work =

A	B	C
7	4	5

Finish =

0	1	2	3	4
false	true	false	true	true

For i = 0 Step 2 ✓

$Need_0$ = 7, 4, 3

Finish [0] is false and $\overset{7,4,3}{Need} < \overset{7,4,5}{Work}$

So P_0 must be kept in safe sequence

Step 3

Work = $\overset{7,4,5}{Work} + \overset{0,1,0}{Allocation_0}$

Work =

A	B	C
7	5	5

Finish =

0	1	2	3	4
true	true	false	true	true

For i = 2 Step 2 ✓

$Need_2$ = 6 , 0, 0

Finish [2] is false and $\overset{6,0,0}{Need_2} < \overset{7,5,5}{Work}$

So P_2 must be kept in safe sequence

Step 3

Work = $\overset{7,5,5}{Work} + \overset{3,0,2}{Allocation_2}$

Work =

A	B	C
10	5	7

Finish =

0	1	2	3	4
true	true	true	true	true

Step 4

Finish [i] = true for $0 \leq i \leq n$

Hence the system is in Safe state

The safe sequence is P_1, P_3, P_4, P_0, P_2

(C) What will happen if process P_1 requests one additional instance of resource type A and two instances of resource type C?

$$\mathrm{Request}_1 = \begin{array}{ccc} A & B & C \\ 1, & 0, & 2 \end{array}$$

To decide whether the request is granted we use Resource Request algorithm

Step 1 ✓

$$\overset{1,\,0,\,2}{\mathrm{Request}_1} < \overset{1,\,2,\,2}{\mathrm{Need}_1}$$

Step 2 ✓

$$\overset{1,\,0,\,2}{\mathrm{Request}_1} < \overset{3,\,3,\,2}{\mathrm{Available}}$$

Step 3

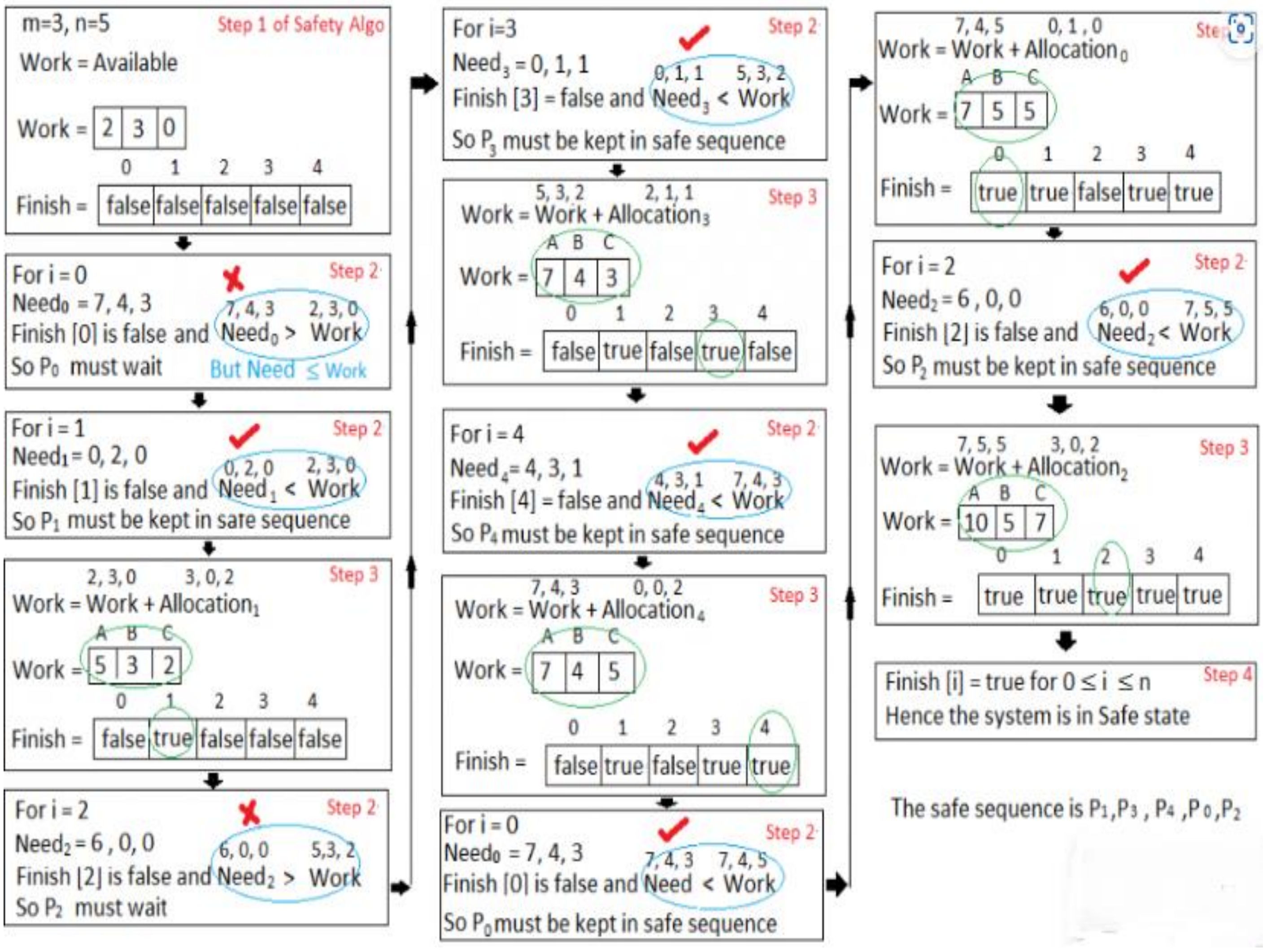
Available = Available – $\mathrm{Request}_1$

$\mathrm{Allocation}_1 = \mathrm{Allocation}_1 + \mathrm{Request}_1$

$\mathrm{Need}_1 = \mathrm{Need}_1 - \mathrm{Request}_1$

Process	Allocation	Need	Available
	A B C	A B C	A B C
P_0	0 1 0	7 4 3	2 3 0
P_1	3 0 2	0 2 0	
P_2	3 0 2	6 0 0	
P_3	2 1 1	0 1 1	
P_4	0 0 2	4 3 1	

We must determine whether this new system state is safe. To do so, we again execute Safety algorithm on the above data structures.



m=3, n=5

Step 1 of Safety Algo

Work = Available

Work =

2	3	0

Finish =

0	1	2	3	4
false	false	false	false	false

For i = 0 — Step 2

$Need_0$ = 7, 4, 3

Finish [0] is false and $\overset{7,4,3}{Need_0} > \overset{2,3,0}{Work}$

So P_0 must wait

But Need ≤ Work

For i = 1 — Step 2

$Need_1$ = 0, 2, 0

Finish [1] is false and $\overset{0,2,0}{Need_1} < \overset{2,3,0}{Work}$

So P_1 must be kept in safe sequence

Step 3

Work = $\overset{2,3,0}{Work}$ + $\overset{3,0,2}{Allocation_1}$

Work =

A	B	C
5	3	2

Finish =

0	1	2	3	4
false	true	false	false	false

For i = 2 — Step 2

$Need_2$ = 6 , 0, 0

Finish [2] is false and $\overset{6,0,0}{Need_2} > \overset{5,3,2}{Work}$

So P_2 must wait

For i=3 — Step 2

$Need_3$ = 0, 1, 1

Finish [3] = false and $\overset{0,1,1}{Need_3} < \overset{5,3,2}{Work}$

So P_3 must be kept in safe sequence

Step 3

Work = $\overset{5,3,2}{Work}$ + $\overset{2,1,1}{Allocation_3}$

Work =

A	B	C
7	4	3

Finish =

0	1	2	3	4
false	true	false	true	false

For i = 4 — Step 2

$Need_4$= 4, 3, 1

Finish [4] = false and $\overset{4,3,1}{Need_4} < \overset{7,4,3}{Work}$

So P_4 must be kept in safe sequence

Step 3

Work = $\overset{7,4,3}{Work}$ + $\overset{0,0,2}{Allocation_4}$

Work =

A	B	C
7	4	5

Finish =

0	1	2	3	4
false	true	false	true	true

For i = 0 — Step 2

$Need_0$ = 7, 4, 3

Finish [0] is false and $\overset{7,4,3}{Need} < \overset{7,4,5}{Work}$

So P_0 must be kept in safe sequence

Step

Work = $\overset{7,4,5}{Work}$ + $\overset{0,1,0}{Allocation_0}$

Work =

A	B	C
7	5	5

Finish =

0	1	2	3	4
true	true	false	true	true

For i = 2 — Step 2

$Need_2$ = 6 , 0, 0

Finish [2] is false and $\overset{6,0,0}{Need_2} < \overset{7,5,5}{Work}$

So P_2 must be kept in safe sequence

Step 3

Work = $\overset{7,5,5}{Work}$ + $\overset{3,0,2}{Allocation_2}$

Work =

A	B	C
10	5	7

Finish =

0	1	2	3	4
true	true	true	true	true

Step 4

Finish [i] = true for $0 \le i \le n$

Hence the system is in Safe state

The safe sequence is P_1, P_3, P_4, P_0, P_2

The system will be in a safe state.