OS

Unit - II

Material

1 Mark Questions

(a) Define Non-Preemptive Scheduling?

→ Here, there is no choice for the Processor in term of Scheduling, it has to take the next instruction

→ Whenever CPU terminates or enter into waiting state, it has to take the new process which is present in the Queue.

(b) Define 'TurnAround Time' (TAT).

→ Its the total time of the process inside the Queue.

(c) Define Mutual Exclusion?

→ Mutual Exclusion means only one process can use a resource at a time.

→ The process can enter into the Critical section and that can be executed.

(d) Define Deadlock? (2.)

→ It is a situation when two process sharing the same resources are effectively preventing each other from accessing the resource.

(e) Define Completion Time.

→ It is the time required for the Process to Complete a Process.

3 Marks

2.

(a) Discuss briefly about 'CPU Scheduling?

→ If there are multiple Process means CPU is scheduled to execute each process one-by one.

→ CPU takes the process that are ready to execute & allocate CPU to one of them.

→ This Short term Scheduler is also known as Dispatcher.

→ Here this Dispatcher execute most frequent process & makes fine grained decision of which process to execute next.

→ CPU Scheduling decisions may take place when a process.

(i.) Switches from running to waiting state.
(ii.) Switches from running to ready state.
(iii.) Switches from waiting to ready state.
(iv.) Terminates.

→ (i.) & (iv.) Comes under Non-Preemptive Scheduling.

→ (ii.) & (iii.) Comes under Preemptive Scheduling.

→ In Non-Preemptive Scheduling, CPU has no choice, it has execute the next process whatever there in the queue.

→ Whereas in Preemptive Scheduling, CPU has a choice of executing any process. based on priority.

(b.) Discuss about Scheduling-Criteria in CPU Scheduling?

→ The CPU Scheduler the processes to execute to schedule the processes to execute and it follows a -Criteria. which is known as the Scheduling-Criteria.

called as Deadlock.

(4.)

→ The Scheduling Criteria is as follows.

(i.) CPU Utilization

Keeps the CPU as busy as possible.

(ii.) Throughput

Number of process that Complete their execution per time unit.

(iii.) Turn Around Time

Amount of time to execute a particular process.

(iv.) Waiting Time

Amount of time a process has been waiting in the Ready Queue.

(v.) ~~Re~~ Response Time

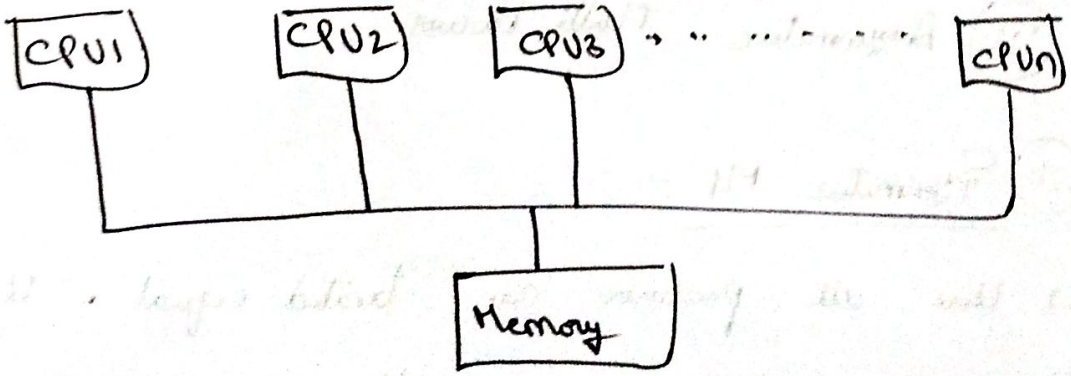
Amount of time it takes from a request was submitted ~~to~~ until the ~~fo~~ first response is produced (not o/p).

Scheduling Algorithm Optimization Criteria (5)

→ Maximum CPU Utilization
→ Maximum Throughput
→ Minimum Turn Around Time
→ Minimum Waiting Time
→ Minimum Reponse Time.

(c) Discuss about Multi-Processing System.

→ If there are more than one processor present in the system which can execute more than one power at the same time.



Multi-Processing System.

→ 2 or more processor with in a single system parallelly.

→ Processor will be sharing memory, I/O devices and resources.

→ Let us Consider 10 processor $P_1, P_2 \ldots\ldots P_{10}$. & if you want to schedule a particular processor first and any processor second -i.e. if you want to execute the processor in a random order.

→ We need to decide the order of processor execution i.e what we need to decide.

→ This comes under Multi-Processing Scheduling.

→ This is of 2 types

(I) Symmetric Multi-Processor

(II) Asymmetric Multi-Processor

(I) Symmetric MP :-

→ Here, all processor are treated equal. Here as there are 10 processor. All processor are treated equal.

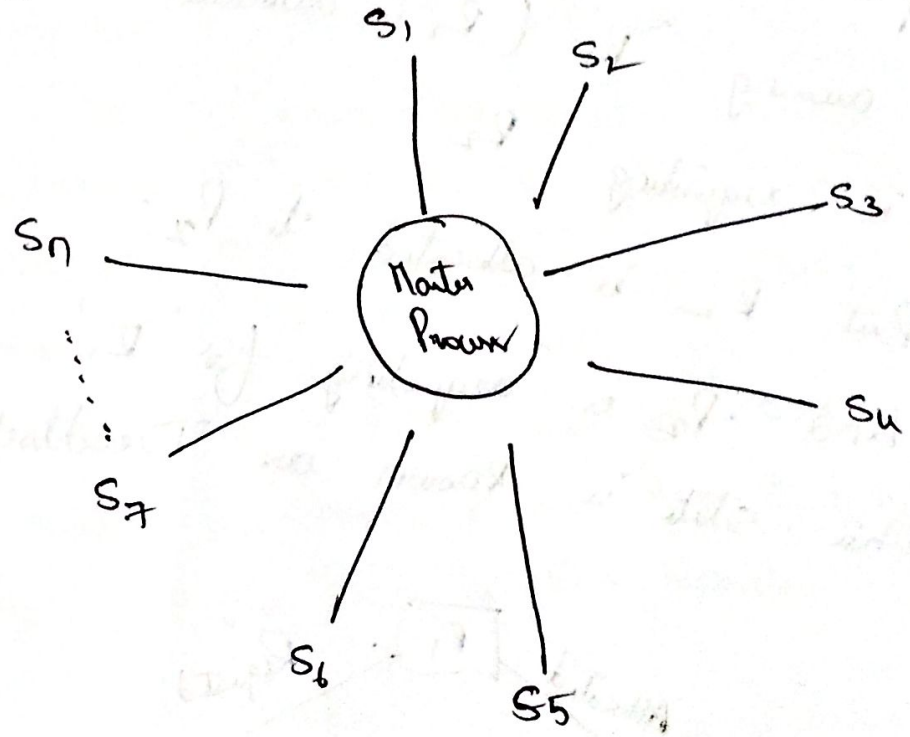
→ One OS Control all CPU's, each CPU has equal right.

→ CPU's are in Peer-to-peer relationship.

(II.) Asymmetric Multi-Processing Syst:

→ Here, Processor are treated as Master & Slave Processor

→ i.e. We have Central Master Processor, this Master Processor will be Co-ordinating all the remaining Processor (Slaves)



Si's → Slave Processor.

→ Advantages

Maximum Throughput, More Reliable, Fast Processing, Efficiency improved & More Economic System.

Note:

→ Throughput: Number of Tasks executed per unit time.

(d.) Discuss about Deadlock. (8.)

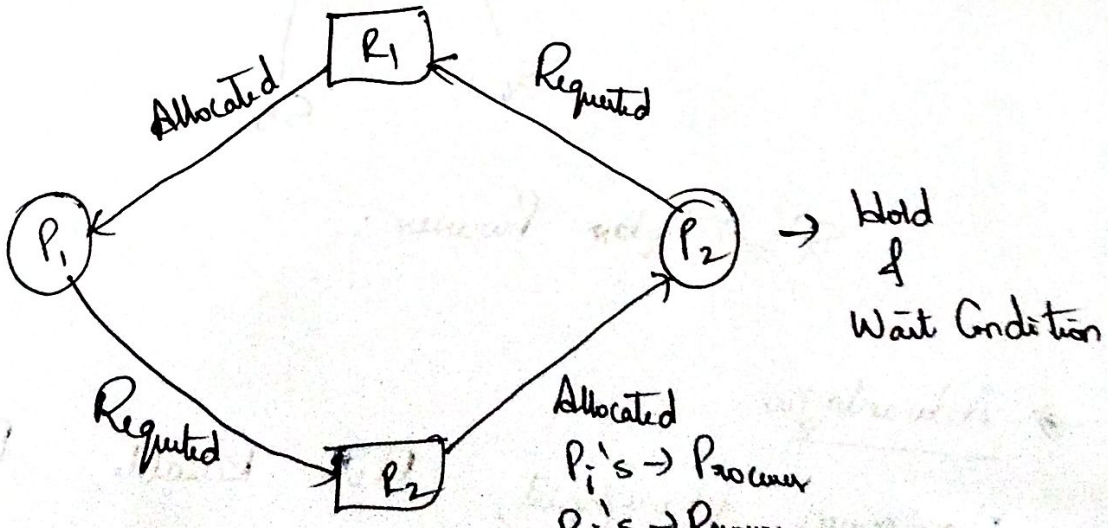
→ It is a situation when 2 process share the same resources are effectively preventing each other from using the resources.

→ Let us Consider two process P_1, P_2 be 2 process & R_1 & R_2 are two resources, P_1 is using R_1 (R_1 is allocated to P_1) but P_1 is requesting R_2.

→ But R_2 is allocated to P_2.

→ And P_2 is requesting for R_1.

→ This state is known as Deadlock.



→ P_1 (Process) is holding R_1 resource & requesting for R_2.

→ P_2 is holding R_2 resource & requesting for R_1.

→ Until P_2 release R_2, P_1 cannot access, until P_1 release R_1, & R_2 cannot access R_1

→ This state is called as Deadlock. (9) (10)

(8) Discuss any general Examples of Deadlock.

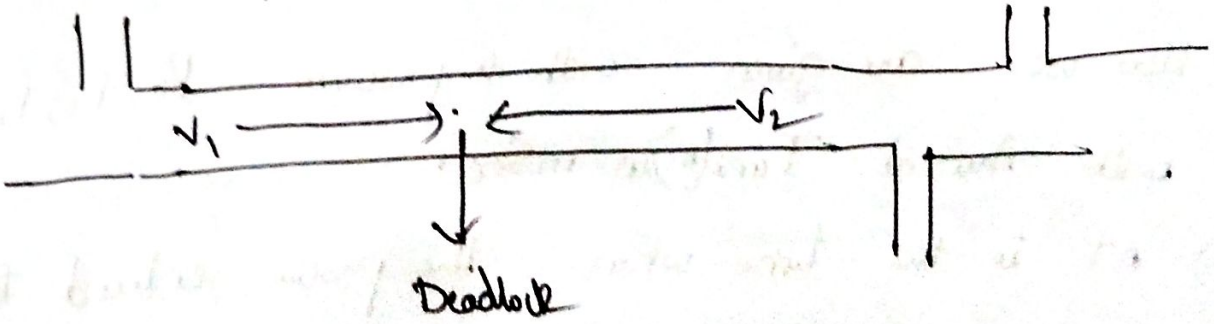
→ Let us Consider an example, Suppose if there 2 friends A & B, if A wants to Call B & B wants to Call A

→ A is Calling B & B is Calling A at same time.

→ So both A, B Calls will get engaged.

→ This situation is Called as Deadlock.

→ Let us take a narrow road, a Vehicle V1 is going & another Vehicle V2 is Coming on the opposite side of V1. There is no space to Cross, Now if V1 or V2 move to Some side road means then there will be some space for the Vehicle to move but here V1 & V2 are not ready to give side to other Vehicles.



5 Marks

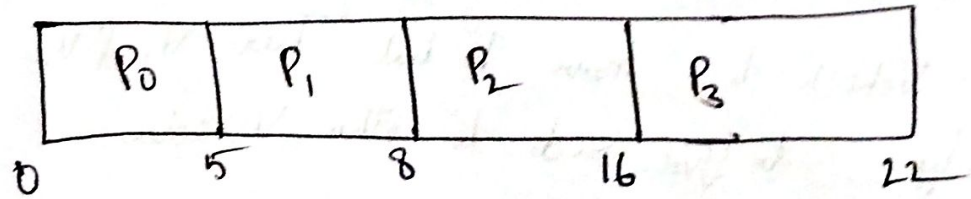
(3) (a) Explain in detail about any CPU Scheduling Technique.

First Come First Serve (FCFS)

→ CPU Scheduling Algorithm defines how CPU Scheduler different jobs & Processes.

Process	Arrival Time (AT)	Burst Time (BT)	Completion Time (CT)	Turn Around Time (TAT)	Waiting Time (WT)
P_0	0	5	5	5	0
P_1	1	3	8	7	4
P_2	2	8	16	14	6
P_3	3	6	22	19	13

Gnatt Chat



→ Here we are given with 4 processes P_0, P_1, P_2 & P_3 with Arrival Time(AT) in mSec.

→ AT is the time when the process entered the Queue.

→ BT is the time required for the processes to Complete itself.

→ Eg: Here if CPU to Complete P_0 means it requires 5mSec. to process it, that is BT

→ Here we need to find out the CT & WT. (11) (15)

→ CT is the Time required for the CPU to Complete a process.

→ TAT is time for which the process is inside the Queue, i.e. the total time of the process inside the Queue.

→ WT is the time for which the process waits for any resource.

→ This is FCFS, . Here the process whatever enters first will be served first., means the first process whatever enter the Queue is executed first.

→ Here for CT of each process, first P_0 arrives at 0 msec & CPU executes the first process first according to FCFS.

→ So p_0 executes at 5msec.

→ Next p_1 enters in to the Queue & the BT of P_1 is 3, So after 5msec when p_0 executes, p_1 will start executing by CPU, So 5+3=8msec, p_1 executes at ~~8sec~~ 8msec.

→ Next P_2 enters the Queue, the BT of p_2 is 8, So after p_1 executes, p_2 starts executed by the CPU, So 8+8 =16msec, So P_2 executes at 16msec

→ Next similarly P3 enter queue, P3 BT is 6, 16+22=22, P3 ~~end~~ execute at 22mSec.

→ So, for TAT, its difference b/w CT & AT (TAT=CT-AT)

TAT(P0) = ~~8-1=7~~ 5-0=5

TAT(P1) = 8-1=7

TAT(P2) = 16-2=14

TAT(P3) = 22-3=19

→ For WT, its difference b/w TAT & BT .i.e.

WT=TAT-BT.

WT(P0) = 5-5=0

WT(P1) = 7-3=4

WT(P2) = 14-8=6

WT(P3) = 19-6=13

→ Next the total TAT is the sum of all TAT's of the processes .i.e.

~~Le~~ Total TAT = 5+7+14+19=45mSec.

→ The Average TAT is given by dividing the Total TAT with number of process i.e.

$$\text{Average (TAT)} = \frac{\text{Total TAT}}{4} = 45/4 = 11.2\text{mSec}.$$

→ The Total WT = 0+4+6+3 = 23mSec.

→ Average (WT) = 23/4 = 5.75mSec.

(b) Explain in detail about Deadlock detection. (13)

→ Resource Allocation Graph (RAG)

→ When a Deadlock occurs, the progress of the system is suspended.

→ Before going to resolving the problem. Let us discuss about Deadlock Detection to know why this Suspension has taken place.

→ Resource Allocation Graph, Resource Allocation means we are allocating the resource to each process.

→ So, how these Resources are allocated to the processes we will arrange Graph for this.

→ Each process utilizes a resource as

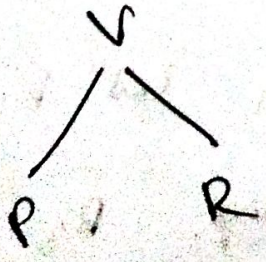
- Request
- Use
- Release

→ This RAG 'G' Consists of

- A set of Vertices V & set of Edges V.

$$G = (V, E)$$

- 'V' is partitioned into 2 types



- P is set of Processes

$P = \{P_1, P_2, \ldots, P_3\}$

the set consisting of all the processes in the system.

- R is set of Resources.

$R = \{R_1, R_2 \ldots R_n\}$

the set of consisting of all the resources in the system.

→ Request Edge is a directed Edge

$P_i \rightarrow R_j$ (P_i requesting for R_j Resource)

→ Assignment Edge is $R_j \rightarrow P_i$

(R_j is assigned to P_i).

→ Now let us Consider an eg for RAG, a resource R_1 with a single instance. Similarly R_3 with single instance; R_2 with 2 instances & R_4 with 3 instances.

→ Instances mean the resources can share to those many processes.

→ A single instance can be shared to a single process.

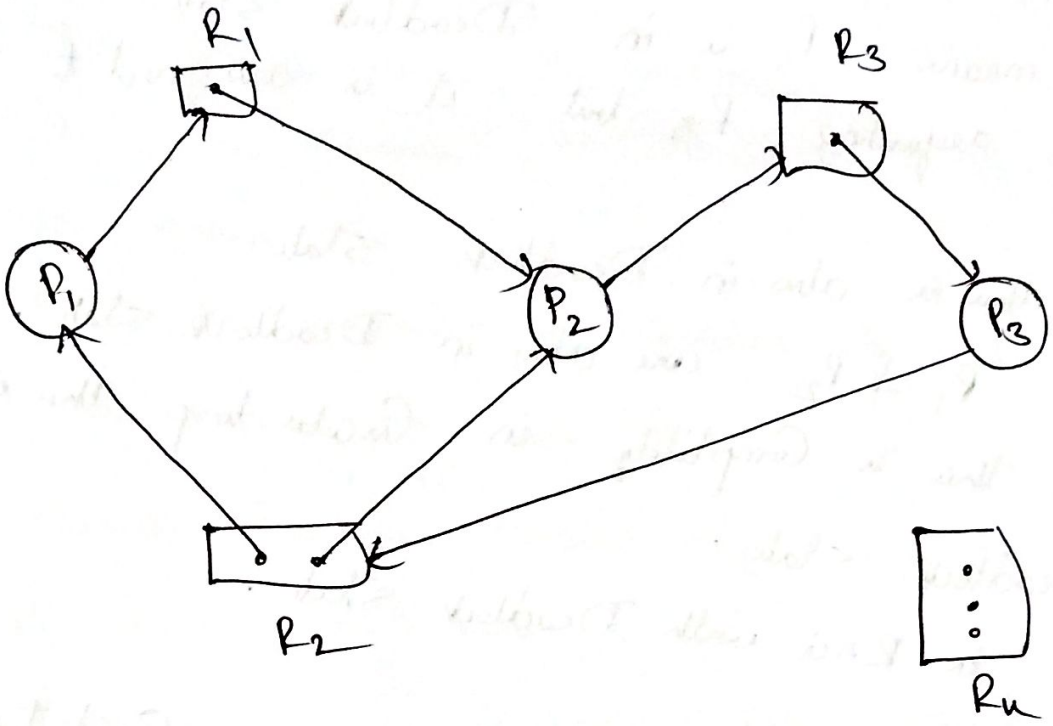
→ Let us Consider we have 3 processes P_1, P_2 & P_3.

→ P_1 is requesting R_1 but R_1 is assigned to P_2.

→ Where P_2 is requesting, R_3 but R_3 is assigned to P_3.

→ One of. instance in R_2 is assigned to P_1 & another instance in R_2 is assigned to P_2.

→ This is the Graphical Representation of RAG.



RAG

→ Here P_1, R_1, P_2 & R_2 form a closed Cycle.

→ ~~P_2, R_3, P_3 & R_n form a Open Cycle.~~

→ Now P_1, R_1, P_2 & R_2 form one Cycle & P_2, R_3, P_3 & R_2 forms another Cycle.

→ So, here a Deadlock may occur, because here P_1 is in blocked state. P_1 is requesting R_1, R_1 is assigned to P_2. So, P_1 need to wait until R_1 is released.

→ One instance of R_2 is given to P_2 & P_3 is requesting for R_2.

→ So, P_3 need to wait until R_2 is released.

→ It means P_3 is in Deadlock state.

→ P_2 is requesting R_3 but it is assigned to P_3.

→ But this is also in Deadlock state.

→ Now P_1 & P_2 are also in Deadlock state.

→ Now this is Completely in Circular loop, this is in Deadlock state.

→ This is RAG with Deadlock state.

(c.) Explain in detail about the Necessary Conditions for Deadlock.

→ Deadlock ~~may~~ may occur when Processes are in

(I.) Mutual Exclusion

(II.) Hold & Wait

(III.) No Preemption

(IV.) Circular Wait

→ When there is Condition ~~any~~ may occur simultaneously for Deadlock to ~~of~~ occur occur

→ So, these 4 Condition may lead to Deadlock.

(I) Mutual Exclusion: (No Sharing of Resources).

→ The Resources are not sharable.

→ If R_1 resource is allocated to P_1 then it is not sharable.

→ If R_1 ~~resource is allocated to~~ it is sharable, any process can access it.

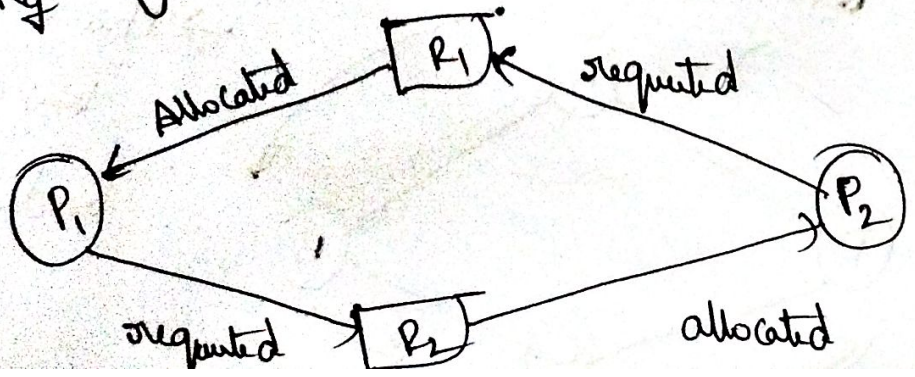
→ Mutual Exclusion means only one process can use a resource at a time.

→ The process can enter into Critical section if that can be executed.

(II) Hold & Wait (P_i's → Processes, R_i's → Resources)

→ P_1 Holding R_1 & waiting for R_2, P_2 holding R_2 & waiting for R_1.

→ Hold & Wait Condition means, a process can holding atleast one resource & will be waiting for another.

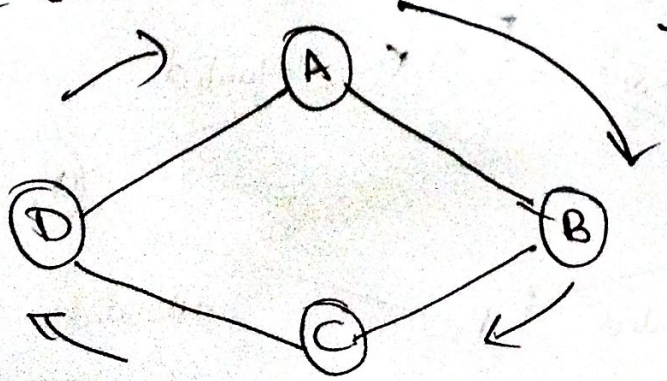


(iii) No Preemption

→ Preemption means if one process completes its resource utilization it can release the resource.
→ So that another process will access that resource.
→ Here a process voluntarily need to release the any resource & there is no other way for any resource to be released.

(iv) Circular Wait

→ Let us consider 4 process A, B, C & D.
→ So, 'A' acquire some resources, which is needed by 'B' & 'B' acquire some resources which need by C.
→ And 'C' acquire another resource which is needed by 'D'.
→ And 'D' acquire some resource which is needed by 'A'.
→ Here everyone is waiting for some resource which is allocated for some other process



→ This is Circular Wait Wait, which leads to Deadlock

(d) Explain in detail about Deadlock Prevention.

→ How to prevent the Deadlock?
→ We need to find a situation before Deadlock occurs.

Necessary Conditions for Deadlock

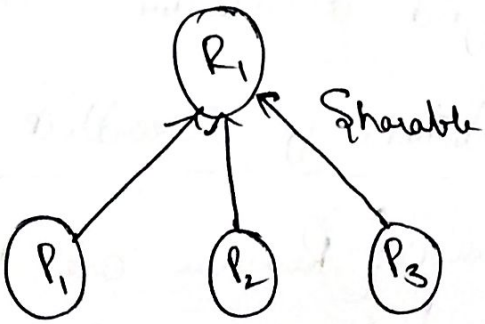
→ Mutual Exclusion (Resources are Non-Sharable)
→ No Preemption (Process need to voluntarily release a Resource)
→ Hold & Wait (Process holding one Resource and waiting for another Resource)
→ Circular Wait (Process waiting for resources which are owned by another process in a Circular order).

→ There are the Condition for Deadlock.
→ If all these Condition are true then Deadlock "may" occur.
→ Deadlock Prevention state that
- if possible try to Prevent all these Conditions or to try to prevent atleast one Condition among all these Condition

Deadlock Prevention → All Condition - FALSE
→ Atleast one Condition - FALSE

→ Make Mutual Exclusion false,
- Just Share the resources, if the resources are sharable, multiple processes then we can avoid Mutual Exclusion

R_i's → Resources
P_i's → Process.



→ Making No Preemption false,
- No Preemption false means making Preemption True.
- This can be done by using time Quantum method.
- Time Quantum method means allocating some time for each process for using any resource.

$P_1(t_1)$ — R_1
$P_2(t_2)$ /

t_i's → Quantum of time

→ Making Hold & Wait false,
- How to make Hold & Wait False?
- By implementing No Hold & Wait
- Just give all the resources to process before it start.
- So it can take any number & can give those resources back.
- This condition is Hold & Wait.

→ Make a Circular Wait False,

- Need to make Circular Wait False.
- By simply given numbering to all the resources, we can prevent Circular Wait.

Eg:- CPU-1, Printer-2, Scanner-3, ...

- So, that the process can request in increasing order.
- Eg:- P_1 is owning 'CPU' & it needs '3' (Scanner), So P_2 which is owning '3', if cannot request for 1 before it is owning 3 it can own for resource greater than 3 & not less than '1'.

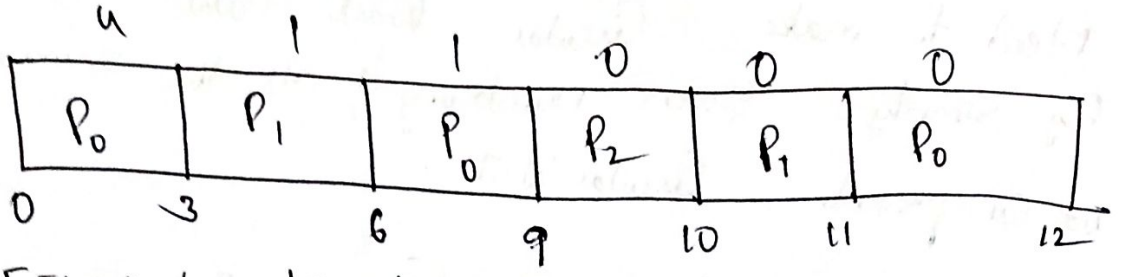
(c) Discuss about Round Robin Scheduling Alg.

→ Here, we have a Criteria called as Quantum (q), 'q' is the maximum time that CPU gives a process at a single point of time. before it will pause the process & move to another process.

→ Here in this case Quantum (q=3)

Process	AT	BT	CT	TAT (CT-AT)	WT TAT-BT
P_0	0	7	12	12	5
P_1	2	4	11	9	5
P_2	4	1	10	6	5

Gnatt Chart



→ First 'p0' enter the queue at 0msec, after 3msec or q=3msec, 'P0' is paced of CPU checks whether any process enter into the queue.

→ So, 2msec, P1 enter the queue, So here the required for p0 to execute is 7-3 = 4msec.

P0 will be in queue after 'P1'.

→ So at 3msec, CPU scheduler 'p1' to execute or q=3 at 3+3=6msec, CPU checks.

→ Now or 'p0' is in the queue after 'p1', CPU checks whether any new process arrived and at 'P2' arrives of P2 will be in the queue after P0 (P0 P1 P0 P2)

→ Now 'p1' is paced by the ~~queue~~ CPU of p0 in queue state ~~executing~~. executing. of 'p1' will be in the queue after P2 (P0 P1 P0 P2 P1)

→ After 3msec, or q=3msec, 6+3=9msec, CPU paces 'p0' of scheduler 'p1' of remaining time needed by 'p0'. to execute in 1msec. (P0 P1 P0 P2 P1 P0).

→ Now p_2 enter the queue, the BT for p_2 is 1m sec, So at 10m sec, p_2 will be executed.

→ Now p_1 enter & execute at ~~time~~ 11m sec, p_1 will be executed.

→ Now finally 'p' enter & execute at 12m sec.

Average (TAT) = 27/3 = 9m sec

Average (WT) = 15/3 = 5m sec.

10 Marks

1.

(a) ~~Dead~~ Explain in detail about Banker's Algorithm.

→ When a process request for any resource it may have to wait.

→ And if process gets all the resources, it need to use those resources & it need to ~~get~~ return them in a finite amount of time.

Eg: P_1 gets 10 & p_2 gets 5 resources, they need to use those resources & need to return them in a finite amount of time.

→ These Conditions need to be taken care when we are dealing with Deadlock Avoidance.

→ First we need to check whether our System is in Safe or Unsafe state.

→ Let us Consider 5 process P_0, P_1, P_2, P_3 & P_4.

→ And a Allocation for each process

A, B, C – 3 Resources.

. Total Count of a resource Can hold

A = 10, B = 5, C = 7

Maximum – Maximum each process Can allocate

Current Work – The Available

(What we need to find out)

Remaining need – Whatever is needed

(Maximum Allocation).

Process	Allocation			Maximum			Current Work (Available)			Remaining need		
	A	B	C	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	3	3	2	7	4	3
P_1	2	0	0	3	2	2	5	3	2	1	2	2
P_2	3	0	2	9	0	2	7	4	3	6	0	0
P_3	2	1	1	4	2	2	7	4	5	2	1	1
P_4	0	0	2	5	3	3	7	5	5	5	3	1
	7	2	5				3	3	2			
							10	5	7			

→ Now P_2 enters the queue, the BT for P_2 is 1msec, so at 10msec, P_2 will be executed.

→ Now P_1 enters & executes at 11msec, P_1 will be executed

→ Now finally P_0 enters & executes at 12msec

$P_0 \rightarrow$ Remaining need (A)

$= Max(P_0, A) - Allocation(P_0, A)$

$= 7 - 0 = 7$

Similarly B & C

$R.N(B) = Max(P_0, B) - Allocation(P_0, B)$

$= 5 - 1 = 4$

$R.N(C) = Max(P_0, C) - Alloc(P_0, C)$

$= 3 - 0 = 3.$

Similarly Remaining Values

Current Work = Total Allocation Count of each resource − Total Time a resource is Allocated

Current Work(A) = $10 - 7 = 3$

→ Now we need to find out which processes are in safe side in a sequence.

→ Now among all the processes P_0 to P_4,

we need to find out which process is in safe side in a sequence (first safe process to last safe process in sequence).

→ Banker's Alg is having a formular to Calculate this

If Remaining Need (R) ≤ Current Work, then Update

Current Work = Current Work + Allocation

→ For P_0,

Remaing Need (R.N) ≤ Current Work (C.W)

R.N			≤	C.W		
A	B	C		A	B	C
7	4	3	≤	3	3	2

$7 \leq 3$ ⊗, $4 \leq 3$ ⊗, $3 \leq 2$ ⊗

So, P_0 is not in safe side

→ For P_1,

R.N			≤	C.W		
A	B	C		A	B	C
1	2	2		3	3	2

$1 \leq 3$ ✓, $2 \leq 3$ ✓, $2 \leq 2$ ✓

So, P_1 is in safe side

Now Update Current Work of P_1

	A	B	C
C.W (P_1) =	3	3	2
⊕	2	0	0
	5	3	2

< P_1

→ For P_2,

$R.N \leq C.W$

A B C ≤ A B C

6 0 0 ≤ 5 3 2

$6 \leq 5$ ⊗, $0 \leq 3$ ⊗, $0 \leq 2$ ⊗

So, P2 is not in safe side.

→ For P_3,

$R.N \leq C.W$

A B C ≤ A B C

2 1 1 ≤ 5 3 2

$2 \leq 5$ ✓, $1 \leq 3$ ✓, $1 \leq 2$ ✓

So, P3 is in safe side.

Update C.W

C.W = C.W + Allocation

A B C

5 3 2

⊕ 2 1 1

7 4 3

<P1, P3

→ For P4,

$R.N \leq C.W$

A B C ≤ A B C

5 3 1 ≤ 7 4 3

$5 \leq 7$ ✓, $3 \leq 4$ ✓, $1 \leq 3$ ✓

Now Update C.W,

C.W = C.W + Allocation

	A	B	C
	7	4	3
⊕	0	0	2
	7	4	5

< P1, P3, P4

→ Now again we need to check P0 & P2 are P1, P3, P4 are executed and some resources are free.

→ For P0,

R.N ≤ C.W

A	B	C	≤	A	B	C
7	4	3	≤	7	4	5

7 ≤ 7 ✓, 4 ≤ 4 ✓, 3 ≤ 5 ✓

~~< P1, P3, P4, P0~~

→ ~~For~~ P Update C.W, C.W = ~~C.W~~ C.W + Alloc

	A	B	C
	7	4	5
⊕	0	1	0
	7	5	5

< P1, P3, P4, P0

→ For P2,

$$R \cdot N \leq C \cdot W$$

A B C ≤ A B C

6 0 0 ≤ 7 5 5

6 ≤ 7 ✓, 0 ≤ 5 ✓, 0 ≤ 5 ✓

Update C-W, C·W + Alloc

7	5	5
3	0	2
10	5	7

<P1, P3, P4, P0, P2>

→ Now all processes entered into safe side

<P1, P3, P4, P0, P2>

→ Now when we execute the processes in this order then Deadlock does not occur.

(b) Explain in detail about Deadlocks.

- Refer 8 Marks (2)-(d) Answer in page-8

Refer 3 Marks Answer (2)-(c) in page-9

Refer 5 Marks Answer (3)-(b) in page-13

Refer 5 Marks Answer (3)-(c) in page-16

Refer 5 Marks Answer (3)-(d) in page-19

(c) Explain in detail about any 2 CPU Scheduling Techniques? (30)

Refer 5 Marks Answer (3)-(a) in page 9 &

Refer 5 Marks, Answer (3)-(e) in page 21