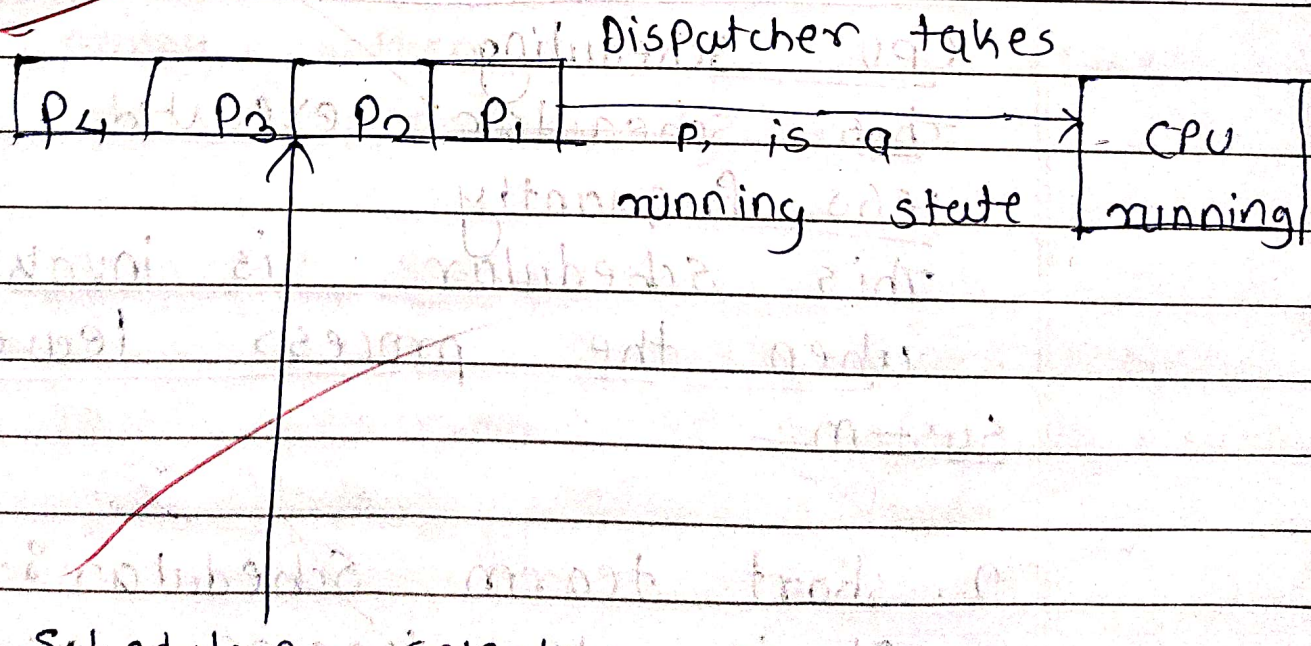
1. What is a dispacher ?

→ The cpu scheduler only selects process to be executed next on a cpu but is cannot assign cpu to the selected processe

- The function of setting up the operating execution of selected process on the cpu is performed by another module of operating system known as dispatcher
- Dispatcher is component which is involves in the cpu scheduling
- This dispatcher needs to be as possible as it is run on every context switch.



Q.2. List the diff type of schedular ? Explain any two type in details?

→ Types of schedular :-

1. long term schedular.
2. short time schedular.
3. medium term schedular.

1] long term schedular :-

It is also known as job schedular. It choose process from the pool (secondary memory) and keep them in ready queue maintained in primary memory. It selected and loads the processes into the memory for execution with the help of CPU scheduling.

This schedular executed the less frequantly

This schedular is invoked when the process leaves system.

2. short tearm schedular :-

It is also known as CPU. Schedular.

Increase the system performance.

This is the changes of ready state to running state of process. It selects process from multiple process that are in ready state in order to execute it and also allocate CPU to one of them.

3] state the benefits of multithreaded programming.

→ Responsiveness :-

Thread management is performed through thread library this library provides function to create and teminate threads to synchronize their activites and to permit them to make request to operating system

Resource sharing :-

Threads shars memory and resources of process to which the belong less address space is require.

Economy :- Threads does not save and restore the execution enviornment. So Context switch time is less

separate processes are not need to create for each request.

Scalability :- In multiprocessor system threads are running in parallel on diff^n processes so multithreading is beneficial.

4] Explain the different types of thread ?

⟶ 1. user level thread :-
They are created and managed by user. They are use at appli-cation level. There is no involvement of OS. The good example is when we use threading in program like Java, C#, python etc.

Program Counter :-
It is responsible for keeping track of instruction and to tell which instruction to execute next.

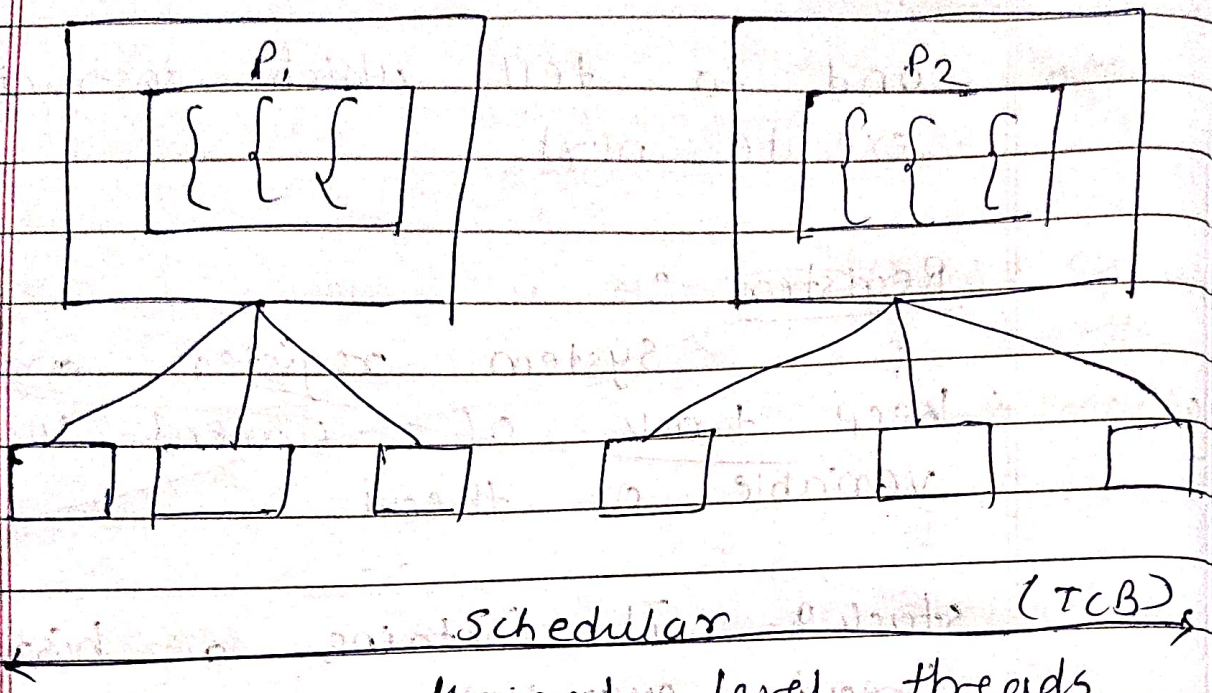
Register :-
System register are their keep track of instruction

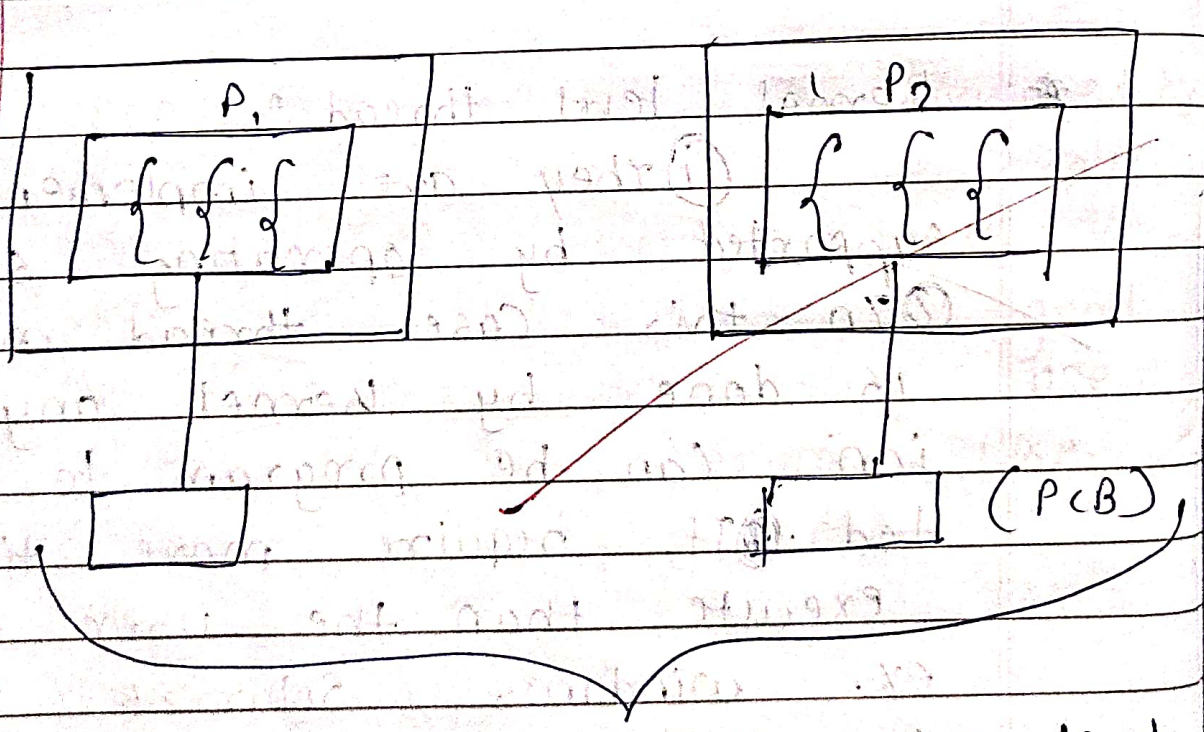
and, to tell which instruction to execute next.

Register :-
System register are their keep track of current working variable of threads.

stack :- It contains a history of thread execution.

Kernel level thread :-
① They are implemented and supported by operating system.
② In this case thread management is done by kernel any application can be program to multithreaded.
③ It require more time to execute than the user thread.
ex. windows, Solarize scheduling by the kernel are done by the thread basis.
The kernal perform the thread creation scheduling and management in kernel space.


P1
P2
Scheduler
(TCB)
Kernal level threads.


P1
P2
(PCB)
Scheduler user level thread.

5] write a short note on PCB.

→ Creating process operating system. maintains process table process. table contains an array of the structure with entry per process its called PCB. Each PCB updated during the transition of process state PCB. include pointer, process id, process. state, process primary register. program counteres, I/O status information and memory limits. The process state is an indicators of the naturs of current activity. in process. the process can be a any one of states like new, Ready, Active, waiting and Halted.

6] write short note on process. termination

→ After execution. Code end, process in terminated. The o.s terminated. process using exit() system call. typically returning int. This init passed along to the. parent if it doing wait() and is typically zero on. successful completion and same.

non-zero code in event of problems.

child code :- int exit code;
exit (exit code);
main()

parent code :- pid_t pid
int status
pid = wait (&status);

when the process is terminated. all process resources - such as memory, I/O files are deallocated or free by O.S.
Parent may terminated, their child rel due to following reasons.

1. child has executed it uses or some resources that it has been allocated.
2. work assign to child no longer requirements.
3. If parent terminates, O.S may not allow child to continue its execution.

Q] Explain diff issue related to message passing system?

→ 1. Synchronization in blocking sends:- In this deadlock and race condition are raised due to waiting processes.

2. Failure handling :- In this network latency message delays and drop system failure affect message passing reliability.

3. Security Concerns :- message passing system there is risk of data interception tampering and unauthorised access in network communication.

4. Buffering :- It many cause prevention of message or the message loss delay or drop due to network latency and process overload.

5. Deadlock and race condition :- This circular dependencies and shared resources access many cause inconsistences and

potential system failure.

@.6 Network latency :- It may lead to delay in a message passing or transmission which affect on the system functionalities.

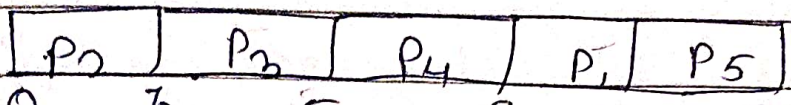
8] Consider the following set of process with the length of the CPU Burst time is given:

Job	Burst Time	Arrival Time
P_1	5	1
P_2	3	0
P_3	2	2
P_4	4	3
P_5	8	4

Solve pre-emtive SJF. Calculate turn around time and waiting time.

→

Gantt chart :-



Average Turn around time.

P1 14 - 1 = 13

P2 3 - 0 = 3

P3 5 - 2 = 3

P4 9 - 3 = 6

P5 22 - 2 = 20

$= \frac{13+3+3+6+20}{5}$

$= \frac{45}{5} = 9$

Average waiting time.

9 - 1 = 8

0 - 0 = 0

3 - 2 = 1

5 - 3 = 2.

14 - 2 = 12

$\frac{8+0+1+2+12}{5}$

$= \frac{23}{5}$

$= 4.6.$

Bajrangbaad 17/8/24

Ass - 2.

Q.1 Explain the rule of wait & signal used in semaphore?

⟹

① wait (P) - it helps to control the entry of the task into the critical section.
- if the value of the wait is positive the the value of the wait argument is decremented in the case of negative or zero value no operation is executed.

e.g. wait (s) - always starting with :
{
while (s2 = 0)
{
no operation)
}
s-- ;
}

② Signal - these type of semaphore operation is used to control the exit of the task from a critical section it helps to increase the value of argument by 1.

e.g. signal (s)
{
=
s++;
=
}

also write q. & answer

Q.2 Describe the cs problem in detail

⟹ - The critical section problem is to design a protocol that the processes can use to co-operate. Each process must request permission to enter it's cs.
- The execution of the cs must be mutually exclusive while one process is in its cs, other process should not be allowed to enter its own critical section.
- every problem should take permission of os to enter its own cs.
- There should be an exit point to the critical section. The balance code after the cs is called os remainder section
- The cs cannot be executed by more than one process at the time.

- If these happens OS faces the difficulties in allowing & disallowing the processes from entering the CS.
  i) mutual exclusion
  ii) progress
  iii) Bounded waiting

remain

9.3 what is process synchronization?

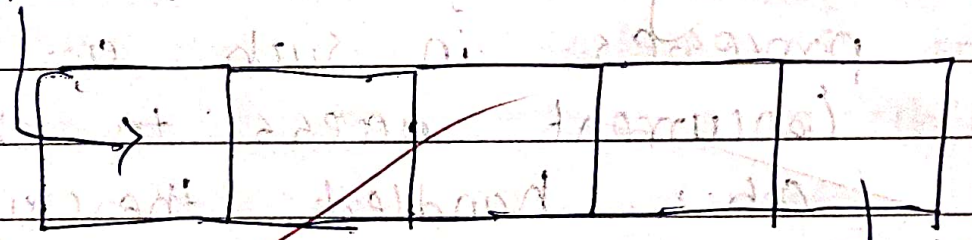
=> - process synchronization also to as process Co-ordination.
- process synchronization means sharing system resources by processes in such a way that concurrent access to shared data is handled thereby minimizing the chance of inconsistant data.
- If there is no controlled access to shared data, it may result in data inconsistency maintaing data Consistency requires machanism to ensure the orderly execution of Co-operating processes.

Q.4. What is Bounded Buffer algorithm.

→ - Bounded - Buffer problem which is also called producer consumer problem. It is one of the classic is a buffer problems of synchronization.

- there is a buffer of n slots & each slot is capable of storing a one unit of data.

- there are 2 processes - producer. (addlineser)



Buffer of slot. → Consumer (access) free

- there will be one of the 3 conditions -

- 1. A producer produces or insert data into a empty slot of the buffer.
- 2. A Consumer remove data from a field slot in the buffer & consume it.

3. A producer should not produces item into the when the consumer is consuming ϕ item from a buffer. ϕ vice versa to the buffer. should only the access by the produces or consumer at a time that need to be a way to make the producer. ϕ consumer work in an independent manner.

Solution: these solution uses the following.

1. mutex :- It is binary semaphore which is used to acquire & release the lav (value=1)

free) 2. empty :- The counting semaphore where initial value is the no of slot in the buffer. since initially all slots are empty so the empty (size of buffer)

3. full :- A counting semaphore whose initial value =0.

Q.5. what is semaphore. Discuss its types.

⇒ - It is technique for synchroniz-ation of a task semaphore will always hold a non-negative integer value.
- A semaphore is a protected variable part from intialisation whose value can be access & attend only wait & signal.

There 2 type of semaphore:.

1) Binary :- If there is only one count of resources value can be '0' or '1' is called as Binary semaphore they are also called as 'mutex law'.

2) Counting semaphore:- Semaphore which as allow arbitory (random) resources count are called as Counting Semaphore the value of these the semaphore is intialized to no:: of resources available. Each process that wishes to use resources perform wait () operation. when process

releases resources it performs a signal().

Q.6. write a short note on Race Condition ?

⟹ i) A situation where several process execute concurrently sharing some data & the result of execution depends on the particular order in which the access to shared data take place is called race condition.

ii) To guard against the race condition we need to ensure than only one process at a time can be manipulating the common resources.

iii) To obtain this, we need some form of process sychronization

iv) the process which completes its task should release the resources & the resources are not available would automatically switch to wait state.

19. What is a reader - writer problem

⇒ i) Reader writer is another classical problem in Concurrent programing. It basically resolves around a no. of processes using a shared global data structure.

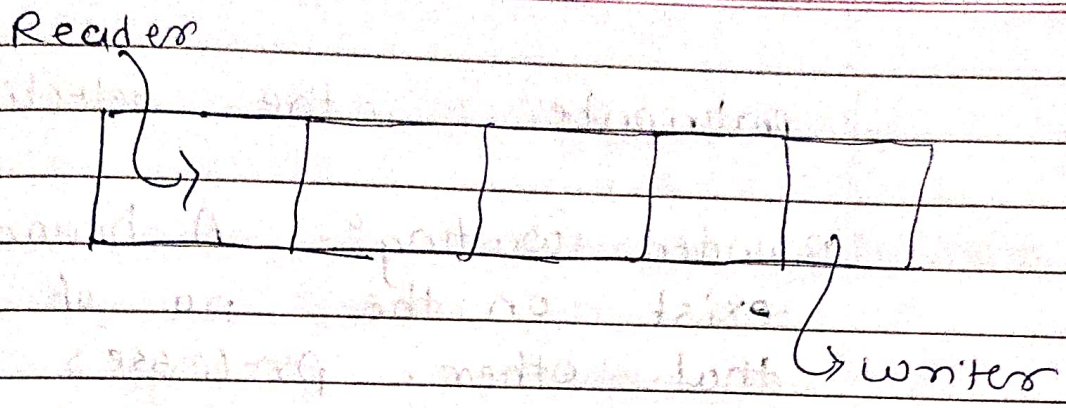
ii) A data set is shared among a number of Concurrent processes. Some processes may want to read the data whereas other may want to write the date.

③ These 2 types of processes are referred as -

i) Reader :- these processes only read the data set, they do not perform any updates.

ii) writer :- these processes can both read & write.

④ The reader writer problem allows multiple reader to read at the same time, but only one single writter can exclusively access the shared data at at the same time such Control can be achieved using semaphore.



Q.8. Describe the requirement of the critical problem solution?

→ The cs problem is to design a protocol that the processes can use to co-operate. Each process must request permission to enter its critical section.

The solution to cs problem must satisfy following requirement.

1. Mutual exclusion :- When one process is in its cs. other process should not be allowed to enter its critical section.

2. Progress :- If no. of process is in cs & there are some process requesting to enter this cs. then all requesting process except the one which is in its remainder section can.

participate in the selection decision.

Bounded waiting:- A bound must exist on the no. of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section & before that request is granted.

gaikwad
22/3/24.

Ass-3.

Q.1 Explain the deadlock relovery in details?

→ When deadlock detection algorithm detects deadlock in system, some relovery scheme must be used to recover the system from deadlock.

1) process termination :-
It can be terminated by two ways :-

i) kill all deadlocked processes? -
This schems is very expensive way of breaking deadlock.

ii) kill one process at a time until deadlock is eliminated: কাজটা হাকল.
This schema involves lots of over head since after killing each process, deadlock detection algorithm must be invoked to the determine whether any processes are still deadlocked.
while killing process one must take care because partial completion of the process may leave the system in inconsistent state.

2. Resource premption :-
In this method successively same resources are preempted & given to other processes unless the deadlock is broken.

i) selecting a victime :-
may be selected based on following criteria :

1. priority :-
Resources of low priority processes will be pre-empted

2. own cost :-
process that has completed very less of its execution will preemtive its resources.

3. Cost affecting other processes :-
How many maximum processes can restart their execution.
If resource of a particular process p are prempted ?

ii) Rollback :-
The process from which resource (or resources are pre-empted will not be able to continue execution such process is rollbacked partially or completely to some safe state

(iii) starvation :-
A process may be held by long time waiting for its resources; then process is said to be the started.
So care must be taken so that. some process is not selected. again & again as victim

Q.2. Difference between deadlock and Starvation.

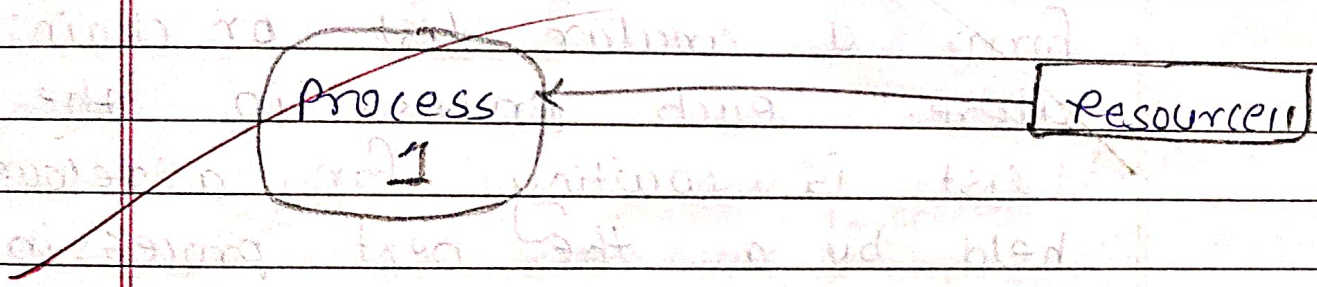
→

	Deadlock	Starvation.
1.	A set of process in deadlock state when every process in set is waiting for an event that can only be caused by another process in the set.	A process may be held for long time waiting for its resources.
2.	All four conditions mutual exclusion. Hold & wait, No pre-emtion & Circular wait.	Due to their priority of process lower priority process selected as victim again &

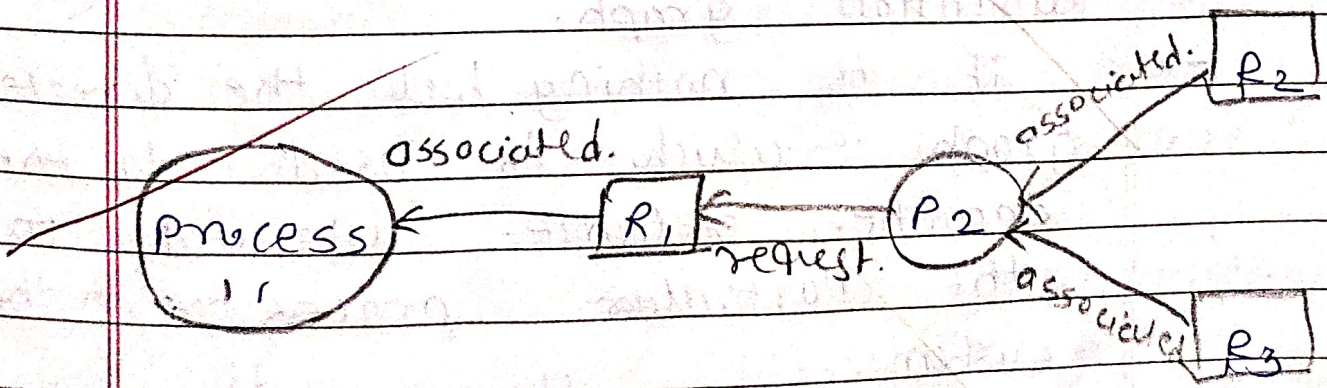
	occur simultenously & then deadlock occurs	again & has to wait for long time. It also known indefinite post-ponement or indefinite blocking or starvation.
3.	It is not possible to have deadlock involving only one single process	It is possible to have starvation involving only one single process.
4.	If deadlock occurs, system cannot progress further since two or more processes may be blocked.	If a starvation occurs system can progress with the other processes.
5.	To avoid deadlock one of method can be implemented: • Deadlock prevention • Deadlock Avoidence • Deadlock detection & recovery	To avoid starvation: • priority of process increased periodically (aging) • count of no. of rollback is kept so that some process should not be selected as victim again

Q3. State the necessary condition for deadlock occur?

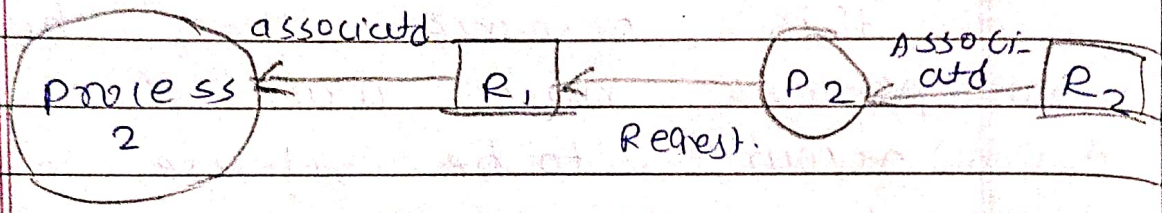
→ 1. mutual exclusion :-
At least one resource must be held in non-sharable mode. If any other processes request this resource then that process must wait for the resource to be release.
their should be resource that can only the held by one processes at a time.



2. Hold & wait :-
A process can be hold multiple resource & still the request move from the other process.



- No- pre-emption :-
  A resource cannot be preempted from process by force.
  A process can only release a resource or resource already to process. can not be pre-empted



4. Circular wait :-
   The processes in the system form a circular list or chain where each process in the list is waiting for a resource held by a the next process in list.

Q.4. Disluss the symbols used for the representation resource allocation graph.

→ It is nothing but the directed graph which is used to the describe resource allocated to the particular proces in the system.

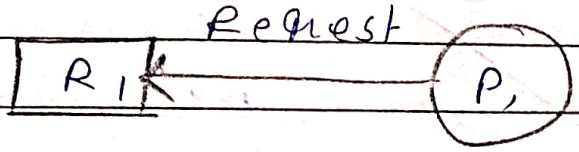
1. process is represented by circle

2. Resource is represented by the rectangle.

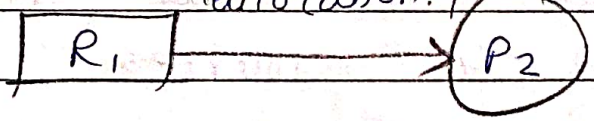
3. The nuber of resources in the similar type (instance) is represented by the dots. The process may be holding the resource or may be requesting for process.

Arrow type :-

1. Request arrow :-



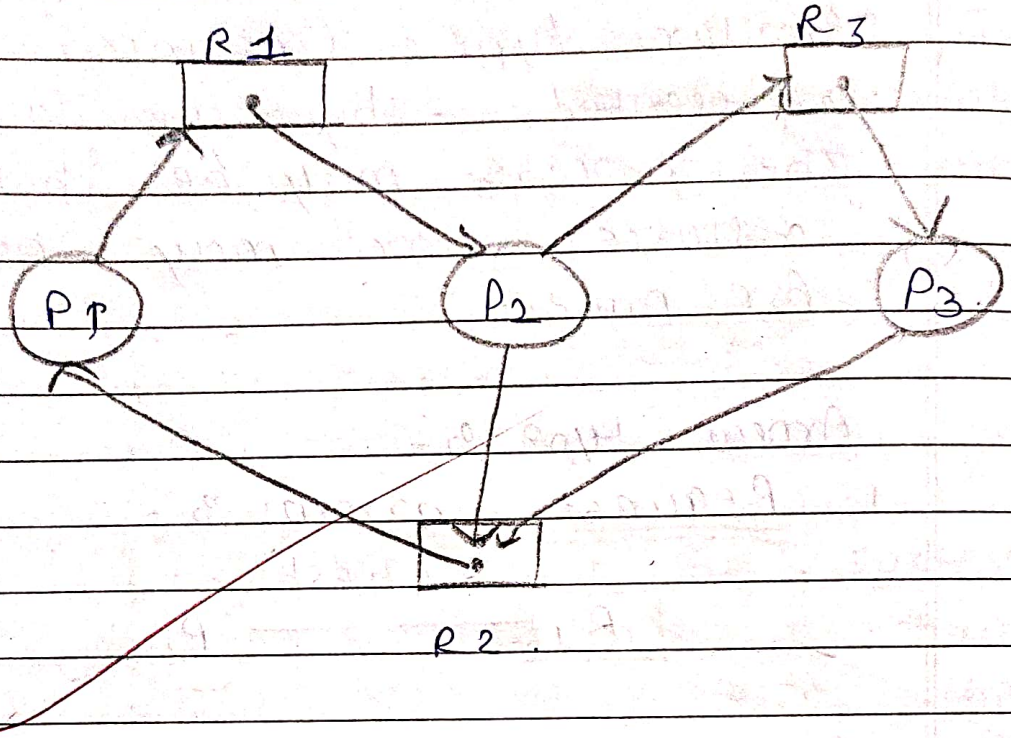
2. Allocation arrow :-



Ex-1. process P_1 is holding an instances of resource type R_2 and it is waiting for an instance of a resource type R_2

2- process P_2 is holding an instance of resource type R_1 & it is the waiting for an instance of a resource type R_3

3. process P_3 is holding an instance of resource type R_3.



Q.5. Consider the given snapshot of system that has 5 process and 4 resources.

Allocation

	A	B	C	D
P_1	0	6	3	2
P_2	0	0	1	2
P_3	1	0	0	0
P_4	1	3	5	4
P_5	0	0	1	4

Max.

	A	B	C	D
P_1	0	6	5	2
P_2	0	0	1	2
P_3	0	7	5	0
P_4	2	3	5	6
P_5	0	6	5	6

Available = A B C D
1 5 2 0