Assignment 5

Deadlock

a) What is meant by deadlock? Illustrate the following deadlock handling techniques: deadlock prevention, deadlock avoidance, and deadlock detection and recovery**.**

Deadlock: A set of blocked processes each holding a resource and waiting to acquire a resource held by another process in the set.

1) Deadlock Prevention:

I. **Mutual Exclusion:** the condition must hold for non-sharable resources. not required for sharable resources.

II. **Hold-and-wait**: must guarantee that whenever a process requests a resource, it does not hold any other resources. Two protocols can be implemented:
   - Require process to request and be allocated all its resources before it begins execution
   - allow process to request resources only when the process has none. A process may request some resources and use them. Before it can request any more resources, it must release all the resources that is currently allocated.
   - Low resource utilization; starvation possible.

III. **No Preemption:** to ensure that this condition does not hold, we can use the following protocol:
   - If a process that is holding some resources requests another resource that cannot be immediately allocated to it, then all resources currently being held are released.
   - Preempted resources are added to the list of resources for which the process is waiting.
   - Process will be restarted only when it can regain its old resources, as well as the new ones that it is requesting.

IV. **Circular Wait:** impose a total ordering of all resource types, and require that each process requests resources in an increasing order of enumeration

2) Deadlock Avoidance:

I. Simplest and most useful model requires that each process declare the maximum number of resources of each type that it may need.

II. The deadlock-avoidance algorithm dynamically examines the resource allocation state to ensure that there can never be a circular-wait condition.

III. Resource-allocation state is defined by:
   - the number of available resources.
   - the number of allocated resources.
   - the maximum demands of the processes.

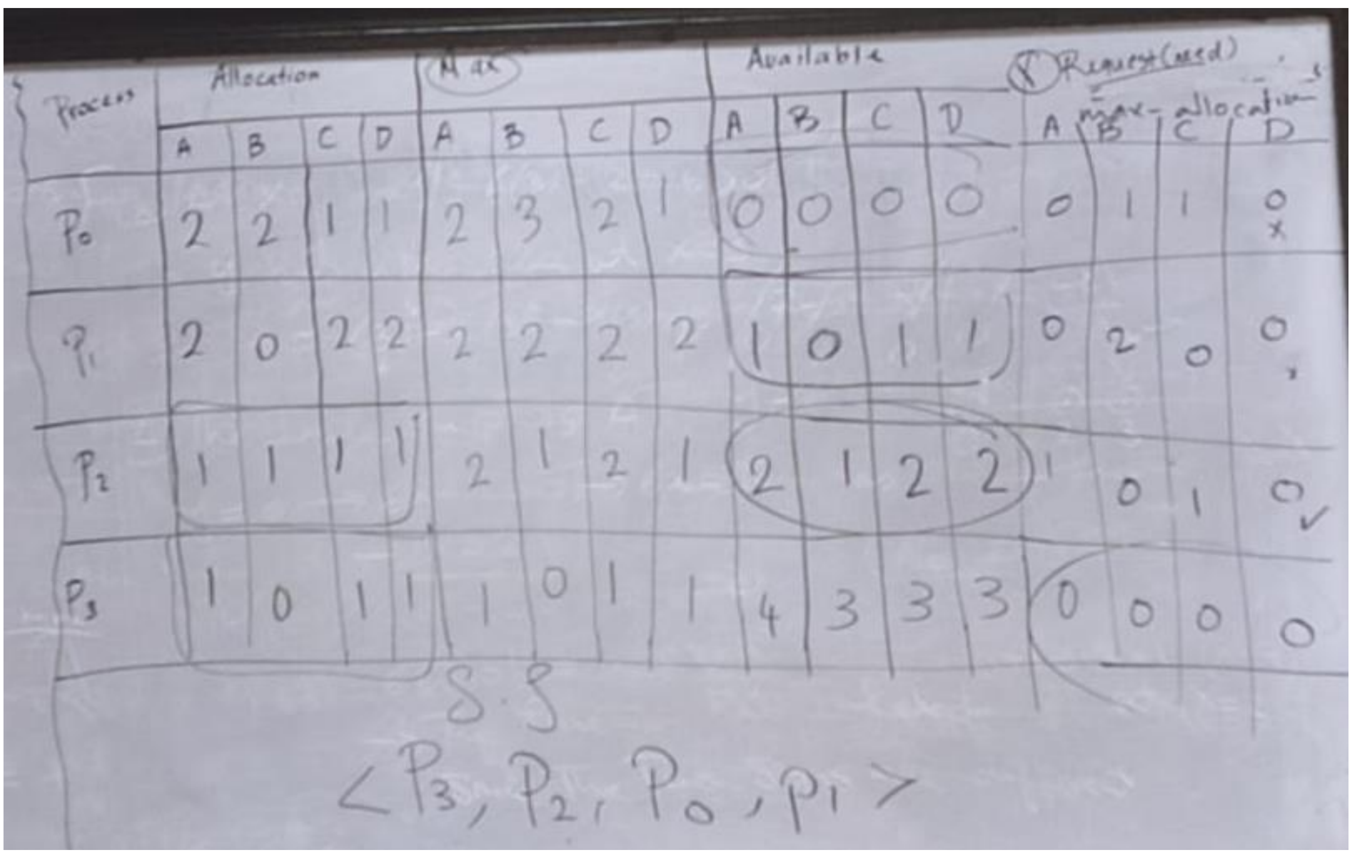
3) Detection and Recovery: Allow process to enter a deadlock state -> detect it -> recover

- When, and how often, to invoke depends on:
  - How often a deadlock is likely to occur?
  - How many processes will be affected by deadlock when it happened and need to be rolled back?

b) Consider the following snapshot of a system:

	Allocation				Max				Available			
	R1	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
P1	2	2	1	1	2	3	2	1	0	0	0	0
P2	2	0	2	2	2	2	2	2				
P3	1	1	1	1	2	1	2	1				
P4	1	0	1	1	1	0	1	1				

Is this system in a safe state? Why? Show your computation step-by-step (using Banker's algorithm). If the system is in a deadlock state, list all processes that involve in a deadlock.



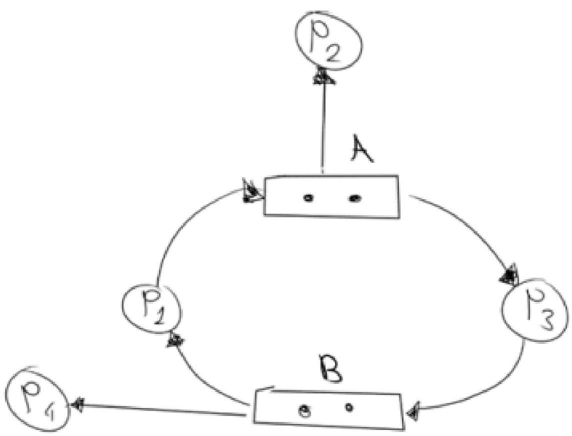
Process	Allocation				Max				Available				Request(need) max - allocation			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
P_0	2	2	1	1	2	3	2	1	0	0	0	0	0	1	1	0 ✗
P_1	2	0	2	2	2	2	2	2	1	0	1	1	0	2	0	0
P_2	1	1	1	1	2	1	2	1	2	1	2	2	1	0	1	0 ✓
P_3	1	0	1	1	1	0	1	1	4	3	3	3	0	0	0	0

S.S

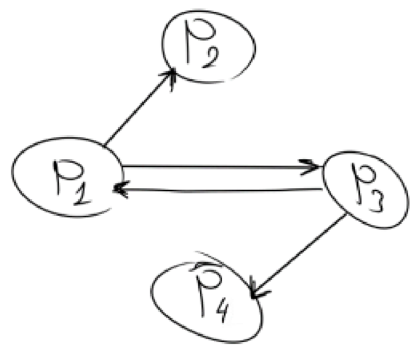
$\langle P_3, P_2, P_0, P_1 \rangle$

c) A system has four processes *P1*, *P2*, *P3* and *P4*, and two resource types *A* and *B*, each of which has two instances. Suppose further *P*1 is requesting an instance of *A* and allocated an instance of *B*; *P*2 is allocated an instance of *A*; *P*3 is requesting an instance of *B* and allocated an instance of *A*; and *P*4 is allocated an instance of *B*. Do the following two problems: **(1)** Draw the resource-allocation graph, and **(2)** Does this system have a deadlock?

Resource allocation graph:



Resource Allocation Graph (RAG)



Wait-For Graph

The system has a loop (A, B). But it is NOT in a deadlock state.

One possible safe sequence = <P4, P2, P1, P3>