Operating System

Unit – 4
(Part-B)
Deadlock

Mayank Mishra
School of Electronics Engineering
KIIT-Deemed to be University

Handling Deadlocks

- Deadlock handling in an operating system refers to the techniques used to prevent, avoid, detect, or recover from situations where processes or threads in the system cannot proceed because they are each waiting for resources that are held by other processes.

- **There are four main strategies for handling deadlocks:**

  - Deadlock Ignorance (Ostrich Method)

  - Deadlock Prevention

  - Deadlock avoidance (Banker's Algorithm)

  - Deadlock detection & recovery

Deadlock Ignorance

- ❑ In the Deadlock ignorance method the OS acts like the deadlock never occurs and completely ignores it even if the deadlock occurs. This method only applies if the deadlock occurs very rarely.
- ❑ The algorithm is very simple. It says, ” if the deadlock occurs, simply reboot the system and act like the deadlock never occurred.” That’s why the algorithm is called the **Ostrich Algorithm**.

Advantages

- Ostrich Algorithm is relatively easy to implement and is effective in most cases.
- It helps in avoiding the deadlock situation by ignoring the presence of deadlocks.

- **Disadvantages**
- Ostrich Algorithm does not provide any information about the deadlock situation.
- It can lead to reduced performance of the system as the system may be blocked for a long time.

Deadlock Prevention

- This strategy attempts to prevent deadlock from occurring in the first place by ensuring that at least one of the necessary conditions for a deadlock is never met. The four necessary conditions for a deadlock are:
    - **Mutual Exclusion:** A resource can be used by only one process at a time. If another process requests for that resource, then the requesting process must be delayed until the resource has been released.
    - **Hold and wait:** Some processes must be holding some resources in the non-shareable mode and at the same time must be waiting to acquire some more resources, which are currently held by other processes in the non-shareable mode.
    - **No pre-emption:** Resources granted to a process can be released back to the system only as a result of voluntary action of that process after the process has completed its task.
    - **Circular wait:** Deadlocked processes are involved in a circular chain such that each process holds one or more resources being requested by the next process in the chain.

Deadlock Prevention (Contd.)

Strategies for Prevention:

- ❑ **Breaking Mutual Exclusion**: Some resources can be shared (e.g., read-only files) so that deadlocks do not occur. In general, however, we can not prevent deadlocks by denying the mutual-exclusion condition, because some resources intrinsically non-sharable.

- ❑ **Preventing Hold and Wait**: To ensure that the hold-and-wait condition never occurs in the system, we must guarantee that, whenever a process requests a resource, it does not hold any other resources.

Deadlock Prevention (Contd.)

Strategies for Prevention:

- **Preventing Hold and Wait**: To ensure that the hold-and-wait condition never occurs in the system, we must guarantee that, whenever a process requests a resource, it does not hold any other resources.
  - One protocol that can be used requires each process to request and be allocated all its resources before it begins execution.
  - An alternative protocol allows a process to request resources only when it has none. A process may request some resources and use them. Before it can request any additional resources, however, it must release all the resources that is currently allocated.

  - Both the above protocols have two main disadvantages:
    - Resource utilization may be low
    - Starvation is possible

Deadlock Prevention (Contd.)

Strategies for Prevention:

- **Enabling Preemption**: Allow resources to be taken from a process if needed, and then resume the process later.
  - If a process that is holding some resource, requests another resource that can not be immediately allocated to it, all resources currently being held are released and if necessary, request again together with the additional resource.
  - If a process requests a resource that is currently held by another process, the OS may pre-empt the second process and require it to release its resources. This works only if both processes do not have the same priority.
  - ✓ This protocols is often applied to resources whose state can be easily saved and restored later, such as CPU registers and memory space. It can not be generally applied to such resources as printers and tape drivers.

Deadlock Prevention (Contd.)

Strategies for Prevention:

- **Breaking Circular Wait**: A way to ensure that this condition never holds is to impress a total ordering of all resource types and to require that each process requests resources in an increasing order of enumeration.

  - Example: Suppose Process P1 is allocated Resource R5. Now if P1 requests for Resources R4 and R3 (which are less than R5), such request will not be granted. Only request for resource greater than R5 will be granted.

  - ✓ Developing an ordering, or hierarchy, in itself does not prevent deadlock. It is up to application developers to write programs that follow the ordering.

Deadlock Avoidance (Banker’s Algorithm)

- Banker’s Algorithm is a resource allocation and deadlock avoidance algorithm used in operating systems. It ensures that a system remains in a safe state by carefully allocating resources to processes while avoiding unsafe states that could lead to deadlocks.

- The Banker’s Algorithm is a smart way for computer systems to manage how programs use resources, like memory or CPU time.

- It helps prevent situations where programs get stuck and can not finish their tasks. This condition is known as deadlock.

- By keeping track of what resources each program needs and what’s available, the banker algorithm makes sure that programs only get what they need in a safe order.

Deadlock Avoidance (Banker's Algorithm)

Components of the Banker's Algorithm:

- ❑ **Available**: This is an array that tells us how many instances of each resource are available.
- ❑ **Maximum**: This is a matrix that tells us the maximum demand of each process for each resource.
- ❑ **Allocation**: This is a matrix that shows the number of resources currently allocated to each process.
- ❑ **Need**: This matrix shows the remaining resources a process needs in order to complete its execution.

$$\boxed{\text{Need} = \text{Maximum} - \text{Allocation}}$$

Deadlock Avoidance (Banker’s Algorithm)

Why Banker’s Algorithm is Named so?

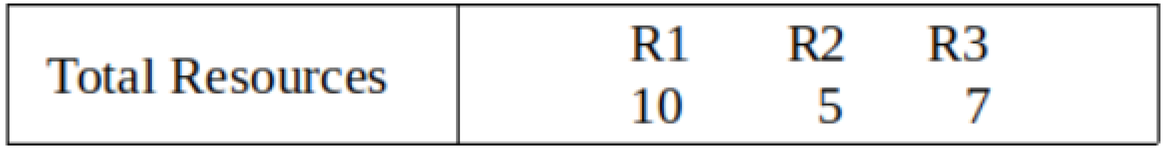
- ❑ The algorithm gets its name from a metaphor where the system is like a banker who must decide whether to lend out resources to processes without putting the system into an unsafe state.

- ❖ The banker’s algorithm is named so because it is used in the banking system to check whether a loan can be sanctioned to a person or not.

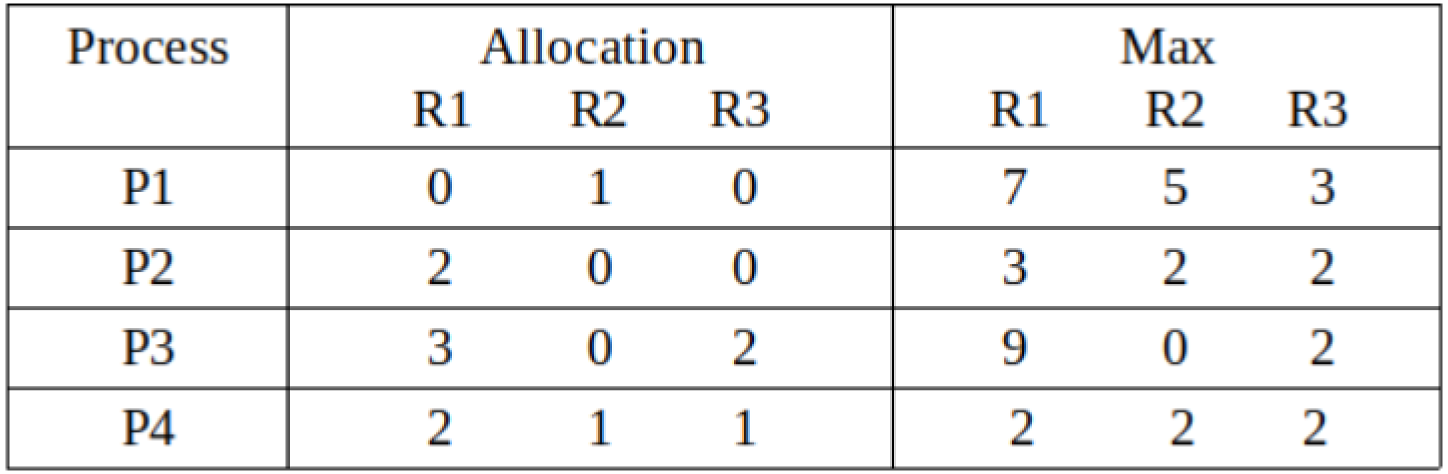
- ❖ Suppose there are n number of account holders in a bank and the total sum of their money is S. Let us assume that the bank has a certain amount of money Y. If a person applies for a loan then the bank first subtracts the loan amount from the total money that the bank has (Y) and if the remaining amount is greater than S then only the loan is sanctioned. It is done because if all the account holders come to withdraw their money then the bank can easily do it..

Deadlock Avoidance (Banker’s Algorithm)

Example:



Total Resources	R1	R2	R3
	10	5	7



Process	Allocation			Max		
	R1	R2	R3	R1	R2	R3
P1	0	1	0	7	5	3
P2	2	0	0	3	2	2
P3	3	0	2	9	0	2
P4	2	1	1	2	2	2

Does Deadlock will occur?

Deadlock Avoidance (Banker's Algorithm)

Total Resources	R1	R2	R3
	10	5	7

Process	Allocation			Max		
	R1	R2	R3	R1	R2	R3
P1	0	1	0	7	5	3
P2	2	0	0	3	2	2
P3	3	0	2	9	0	2
P4	2	1	1	2	2	2

Process	Max			Allocation			Available			Needed		
	R1	R2	R3	R1	R2	R3	R1	R2	R3	R1	R2	R3
P1	7	5	3	0	1	0	3	3	4	7	4	3
P2	3	2	2	2	0	0				1	2	2
P3	9	0	2	3	0	2				6	0	0
P4	2	2	2	2	1	1				0	1	1
				7	2	3						

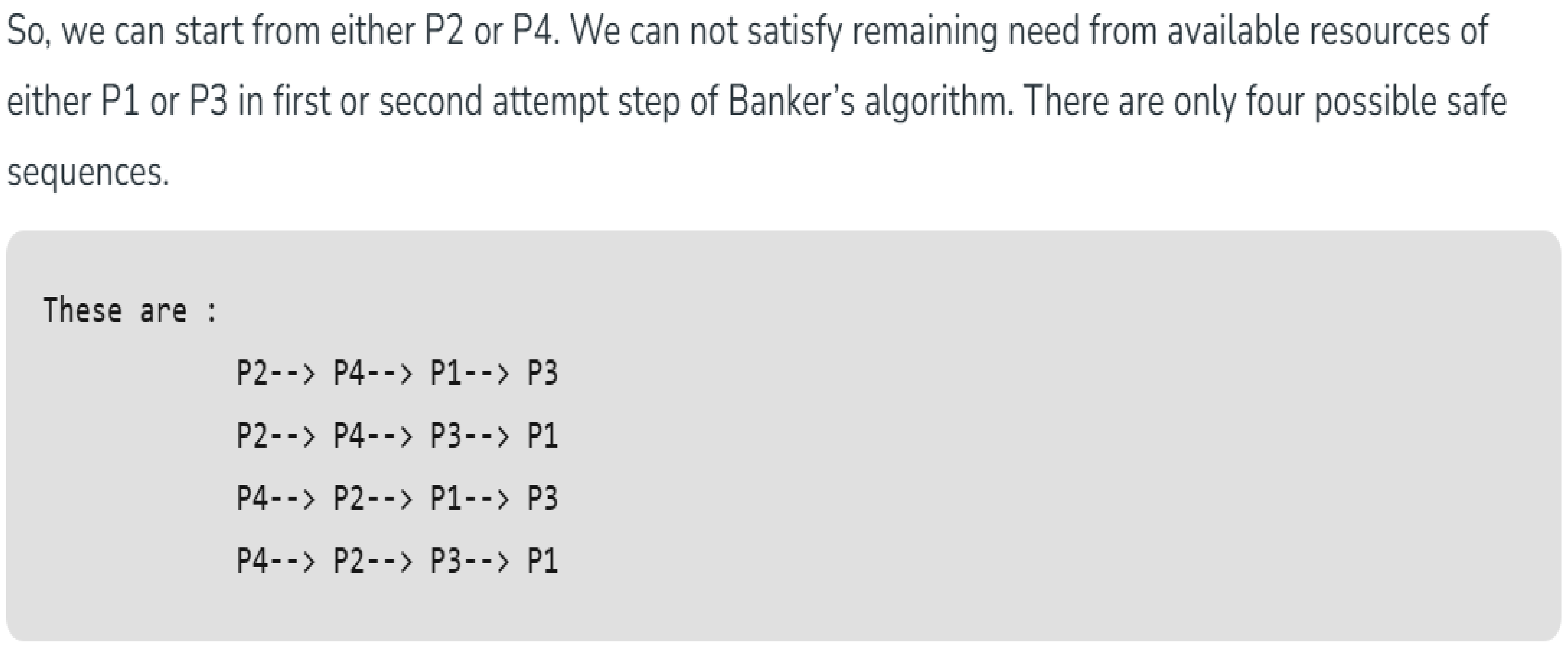
Total resources are R1 = 10, R2 = 5, R3 = 7 and allocated resources are R1 = (0+2+3+2 =) 7, R2 = (1+0+0+1 =) 2, R3 = (0+0+2+1 =) 3. Therefore, remaining resources are R1 = (10 – 7 =) 3, R2 = (5 – 2 =) 3, R3 = (7 – 3 =) 4.

Remaining available = Total resources – allocated resources

and

Remaining need = max – allocated

Process	Max			Allocation			Available			Needed		
	R1	R2	R3	R1	R2	R3	R1	R2	R3	R1	R2	R3
P1	7	5	3	0	1	0	3	3	4	7	4	3
P2	3	2	2	2	0	0				1	2	2
P3	9	0	2	3	0	2				6	0	0
P4	2	2	2	2	1	1				0	1	1
				7	2	3						



So, we can start from either P2 or P4. We can not satisfy remaining need from available resources of either P1 or P3 in first or second attempt step of Banker's algorithm. There are only four possible safe sequences.

```
These are :
          P2--> P4--> P1--> P3
          P2--> P4--> P3--> P1
          P4--> P2--> P1--> P3
          P4--> P2--> P3--> P1
```

No Deadlock

Question 1:

Process	Allocation			Maximum			Available		
	E	**F**	**G**	**E**	**F**	**G**	**E**	**F**	**G**
P0	1	0	1	4	3	1	3	3	0
P1	1	1	2	2	1	4			
P2	1	0	3	1	3	3			
P3	2	0	0	5	4	1			

Does the Deadlock present?

Question 1:
Solution:

Process	Allocation			Maximum			Need		
	E	**F**	**G**	**E**	**F**	**G**	**E**	**F**	**G**
P0	1	0	1	4	3	1	3	3	0
P1	1	1	2	2	1	4	1	0	2
P2	1	0	3	1	3	3	0	3	0
P3	2	0	0	5	4	1	3	4	1

Remaining Need must be less than or equal to Current Availability

Question 1:

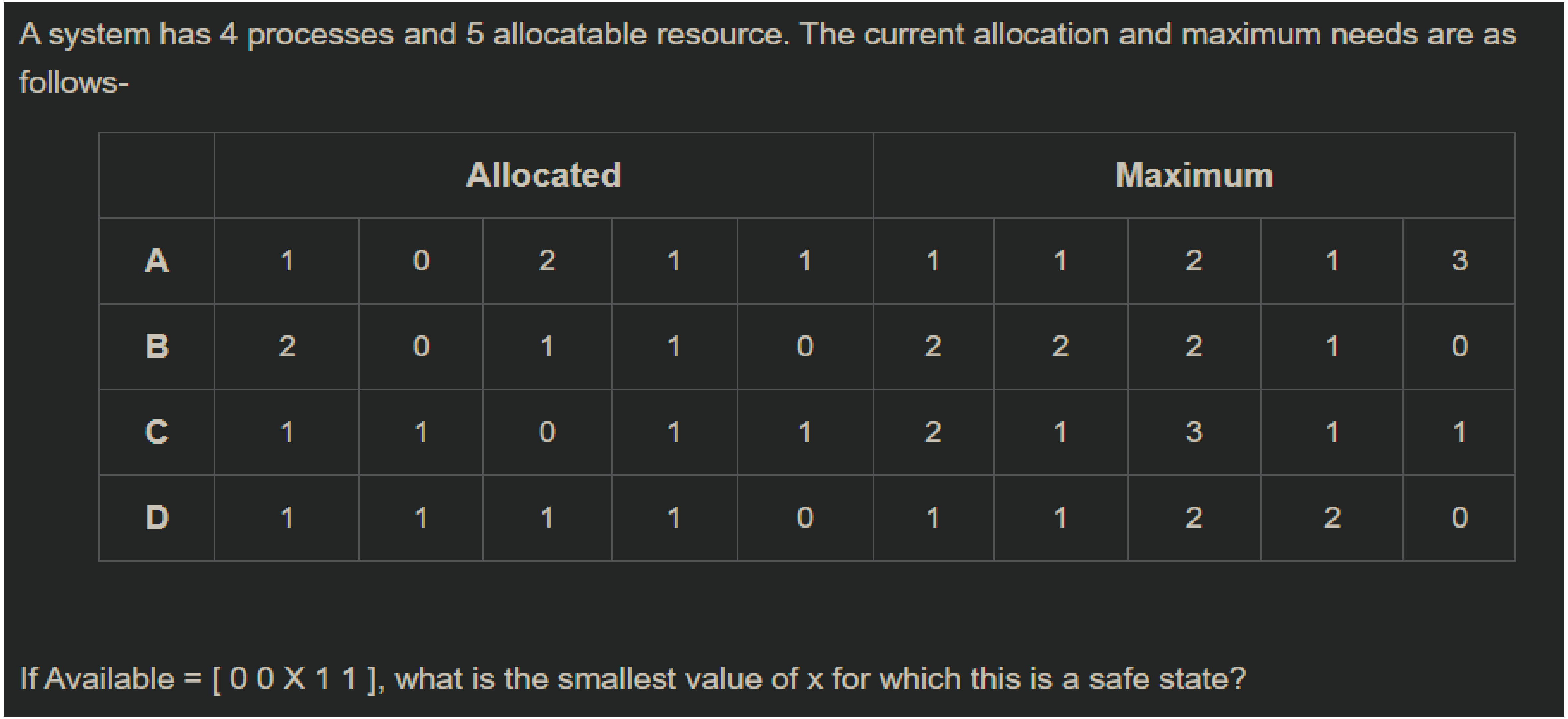
Solution:

Process	Allocation			Maximum			Need		
	E	**F**	**G**	**E**	**F**	**G**	**E**	**F**	**G**
P0	1	0	1	4	3	1	3	3	0
P1	1	1	2	2	1	4	1	0	2
P2	1	0	3	1	3	3	0	3	0
P3	2	0	0	5	4	1	3	4	1

Remaining Need must be less than or equal to Current Availability

One of the safe sequence:P0 – P2- P1 –P3

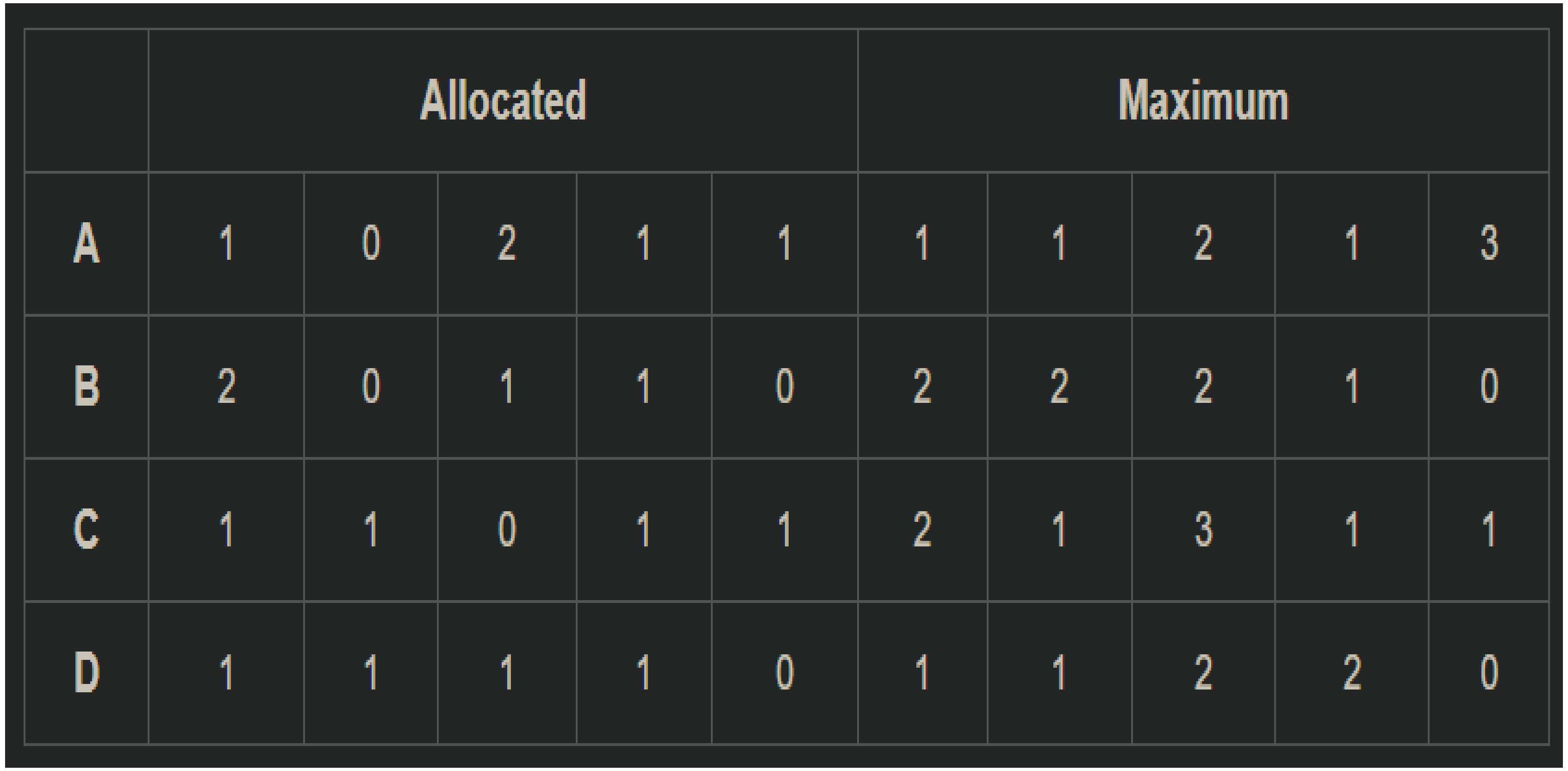
Question 2:



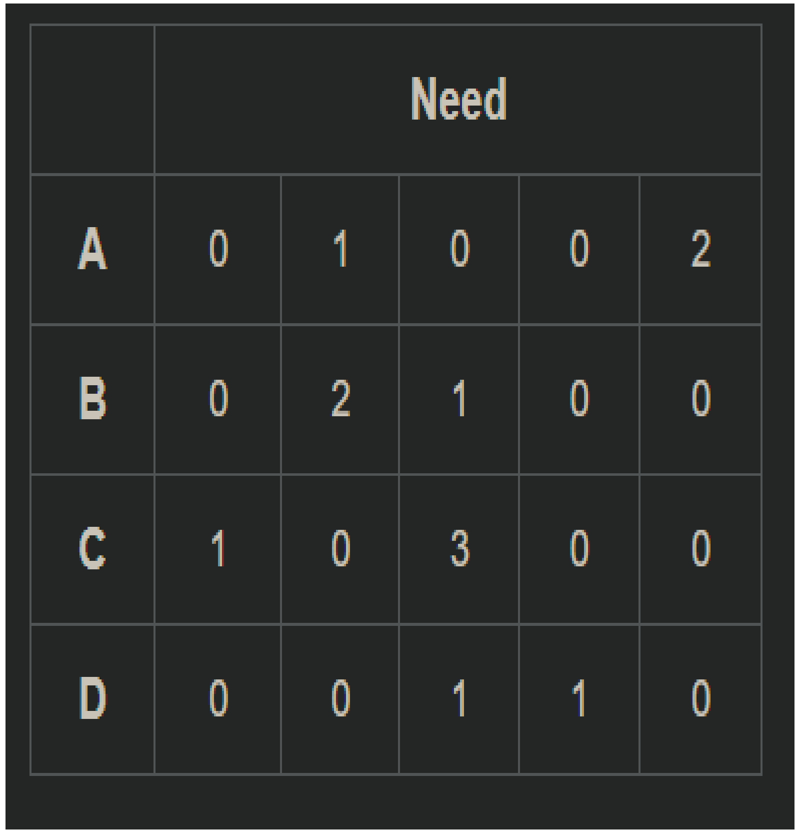
A system has 4 processes and 5 allocatable resource. The current allocation and maximum needs are as follows-

	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?



	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



	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 1: X=0
Deadlock present

- ❑Case 2: X=1
Deadlock present

- ❑Case 3: X=2
No Deadlock

Question 3:

A system is having 3 processes each require 2 units of resources ‘R’. The minimum no. of units of ‘R’ such that no deadlock will occur_______________.

a) 3
b) 5
c) 6
d) 4

Question 3:

Solution:

<u>Case I:- R=2</u>

P1	P2	P3
2	0	0

P1 will be executed and its resources will be released

If

P2	P3
1	1

Deadlock will occur

Question 3:

Solution:

<u>Case II:- R=3</u>

P1	P2	P3
1	1	1

Deadlock will occur

Solution:

<u>Case III:- R=4</u>

P1	P2	P3
2	1	1

Question 3:

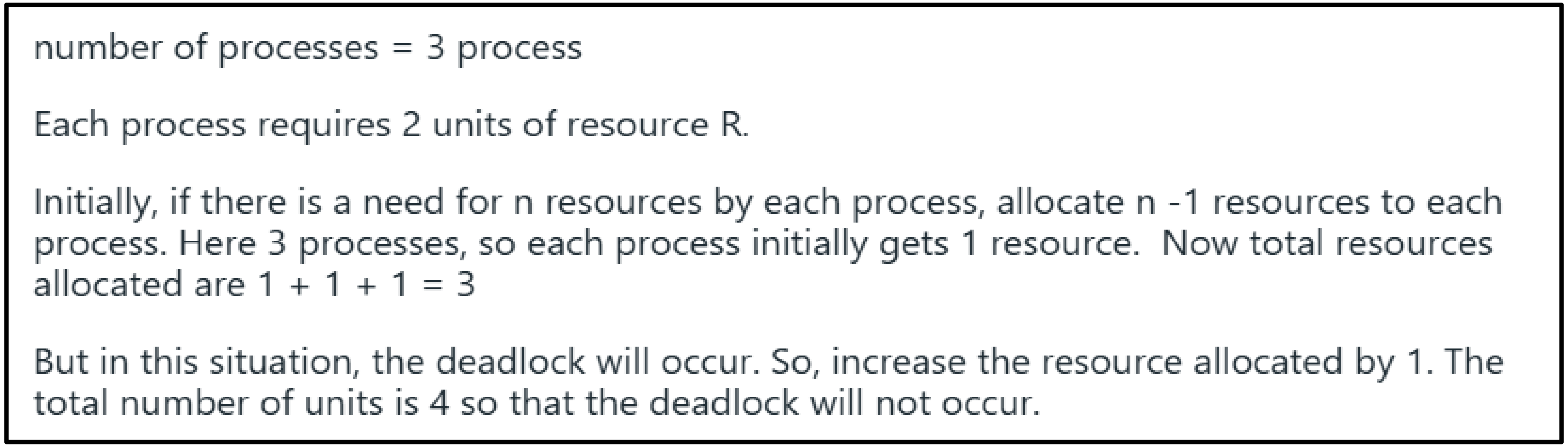
Solution:

Answer: R=4

<u>Case III:- R=4</u>

P1	P2	P3
2	1	1

Deadlock will not occur



number of processes = 3 process

Each process requires 2 units of resource R.

Initially, if there is a need for n resources by each process, allocate n -1 resources to each process. Here 3 processes, so each process initially gets 1 resource. Now total resources allocated are 1 + 1 + 1 = 3

But in this situation, the deadlock will occur. So, increase the resource allocated by 1. The total number of units is 4 so that the deadlock will not occur.

Question 4:

A system is having 10 user processes each requiring 3 units of resource R. The minimum number of units of R such that no deadlock will occur _____?

Question 4:

A system is having 10 user processes each requiring 3 units of resource R. The minimum number of units of R such that no deadlock will occur _____?

Solution:

Minimum number of units of resource R that ensures no deadlock = 20 + 1 = 21

Question 5:

A system is having 10 user processes each requiring 3 units of resource R. Maximum number of units of resource R that ensures deadlock_____?

Question 5:

A system is having 10 user processes each requiring 3 units of resource R. Maximum number of units of resource R that ensures deadlock_____?

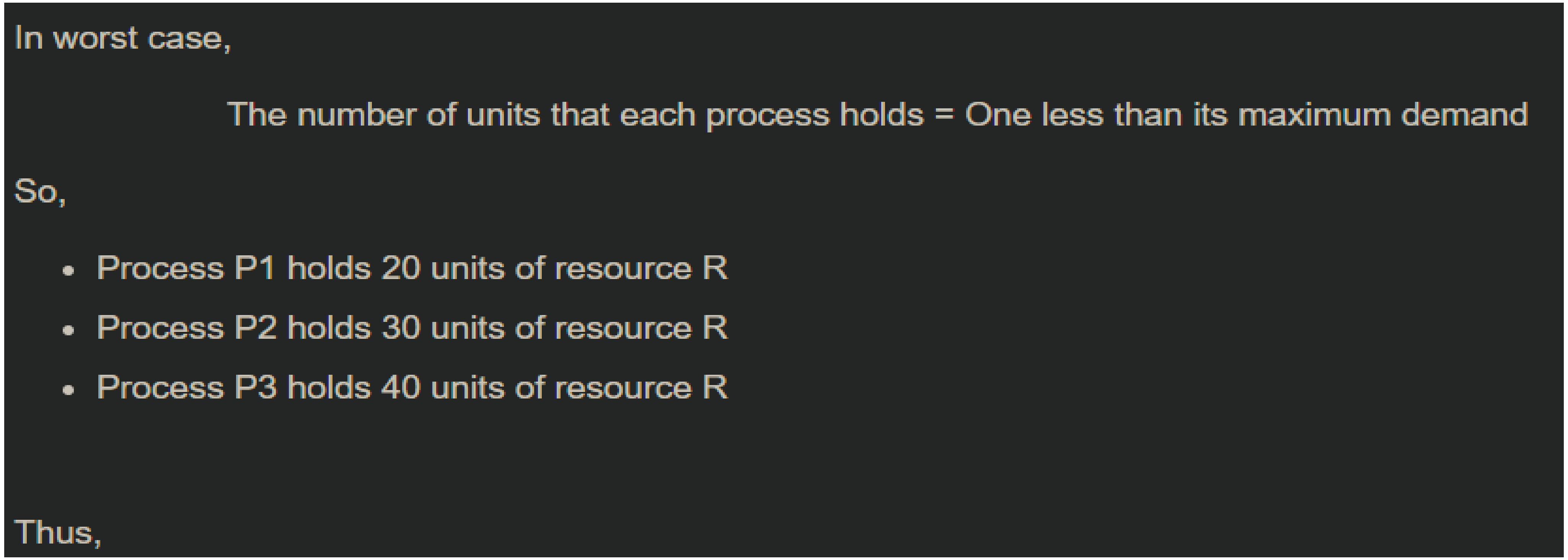
Solution:

Maximum number of units of resource R that ensures deadlock= 20

Question 6:

A system is having 3 user processes P1, P2 and P3 where P1 requires 21 units of resource R, P2 requires 31 units of resource R, P3 requires 41 units of resource R. The minimum number of units of R that ensures no deadlock is _____?

Solution for Question 6:

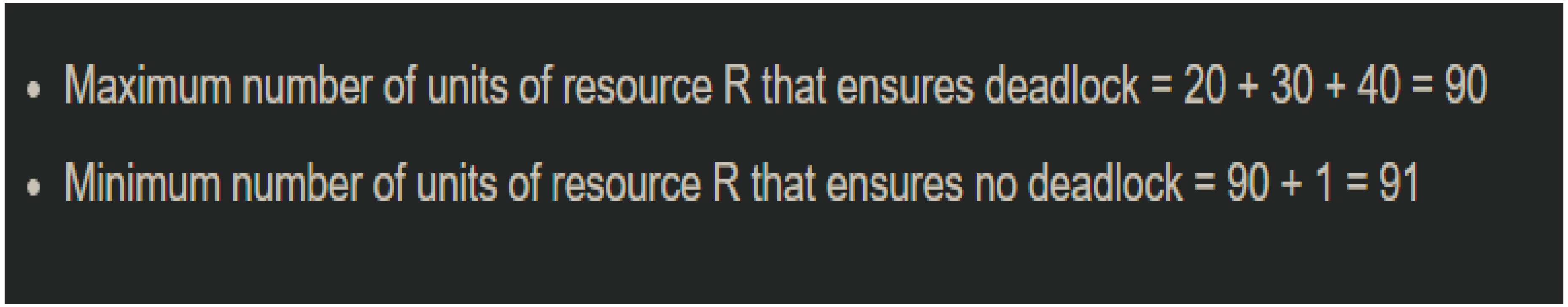

In worst case,

The number of units that each process holds = One less than its maximum demand

So,

- Process P1 holds 20 units of resource R
- Process P2 holds 30 units of resource R
- Process P3 holds 40 units of resource R

Thus,


- Maximum number of units of resource R that ensures deadlock = 20 + 30 + 40 = 90
- Minimum number of units of resource R that ensures no deadlock = 90 + 1 = 91

Question 7:

If there are 100 units of resources R in the system and each process in the system requires 5 units of resource R, then how many processes can be present at maximum so that no deadlock will occur?

a) 23
b) 24
c) 25
d) None

Question 7:

Solution

Let us check the minimum number of process causing Deadlock:

$$\frac{100}{4} = 25 \Rightarrow$$

25 number of minimum process will cause deadlock

Hence, 24 number of maximum process will not result into the Deadlock

Answer: Option B

Question 8:

Consider the Banker algorithm with 4 processes P0, P1, P2, P3 and resources A, B, and C

Process	Maximum			Allocation			Available		
	A	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**
P0	4	1	3	2	1	1	**2**	**0**	**2**
P1	1	2	3	1	1	1			
P2	1	2	3	0	1	1			
P3	2	1	2	0	1	1			

Statement 1: If a process P0 requests (1,0,2), will it be granted?

Statement 2: If a process P2 requests (1,1,0) will it be granted?

Which of the above statements are true?

Question 8:

Solution:

Process	Maximum			Allocation			Need		
	A	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**
P0	4	1	3	2	1	1	2	0	2
P1	1	2	3	1	1	1	0	1	2
P2	1	2	3	0	1	1	1	1	2
P3	2	1	2	0	1	1	2	0	1

As P0 request (1,0,2) it will change the **allocation** of P0 as well as its **need** along with the **current availability** which is (2,0,2).

Question 8:

Solution:

Process	Maximum			Allocation			Need		
	A	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**
P0	4	1	3	3	1	3	1	0	0
P1	1	2	3	1	1	1	0	1	2
P2	1	2	3	0	1	1	1	1	2
P3	2	1	2	0	1	1	2	0	1

Current availability = (1,0,0).

Now check whether Deadlock is present or not if P0 request is granted!!

Question 8:

Solution:

Process	Maximum			Allocation			Need		
	A	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**
P0	4	1	3	3	1	3	1	0	0
P1	1	2	3	1	1	1	0	1	2
P2	1	2	3	0	1	1	1	1	2
P3	2	1	2	0	1	1	2	0	1

One of the safe sequence:P0 – P1- P2 –P3

Similarly check for Statement 2 from starting

Similarly check for Statement 2: If a process P2 requests (1,1,0) will it be granted?

Similarly check for Statement 2: If a process P2 requests (1,1,0) will it be granted?

Deadlock is present for Statement 2

Final Answer: Statement 1 is TRUE
Statement 2 is FALSE

References

1. Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, “Operating System Concepts,” Eleventh Edition (Willey).

2. Andrew S. Tanenbaum, “Modern Operating Systems”, Fourth Edition (Pearson Publications), 2014.

3. https://www.geeksforgeeks.org/

4. https://www.javatpoint.com/

5. https://www.tutorialspoint.com/

6. https://www.nesoacademy.org/

7. https://www.baeldung.com/

8. https://www.educative.io/

9. https://prepinsta.com