Register Number | | | | | | | | | |

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110

(An Autonomous Institution, Affiliated to Anna University, Chennai)

Department of Computer Science and Engineering

Continuous Assessment Test – I

Answer Key

Degree & Branch	BE & Computer Science and Engineering				**Semester**	VII
Subject Code & Name	UCS1701- Distributed Systems				**Regulation:**	**2018**
Academic Year	2023-2024 ODD	**Batch**	2020-2024	**Date**	**05.09.2023**	**FN / ~~AN~~**
Time: 08:10 - 09:40 A.M (90 Minutes)	**Answer All Questions**				**Maximum: 50 Marks**	

COURSE OUTCOMES

CO1- Realize the foundations of Distributed Systems [K2]
CO2- Able to solve synchronization and state consistency problems [K3]
CO3- Demonstrate the resource sharing techniques in Distributed systems [K3]
CO4- Comprehend the working model of consensus and reliability of Distributed Systems [K3]
CO5- Identify the fundamentals of Peer-to-Peer Systems [K2].
CO6- Formulate a synchronization problem for an ad-hoc distributed system and adapt its solution [K6]

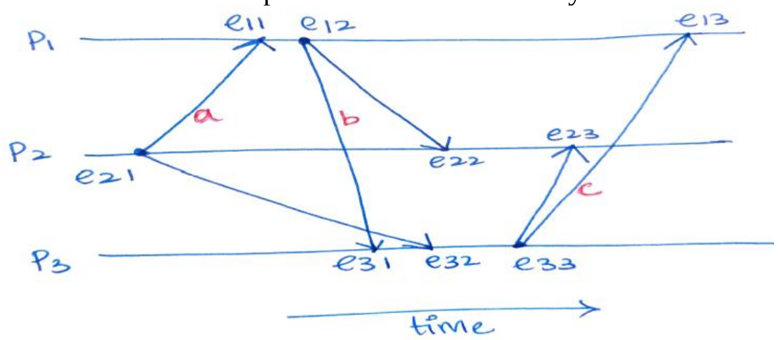
Part – A (6×2 = 12 Marks)

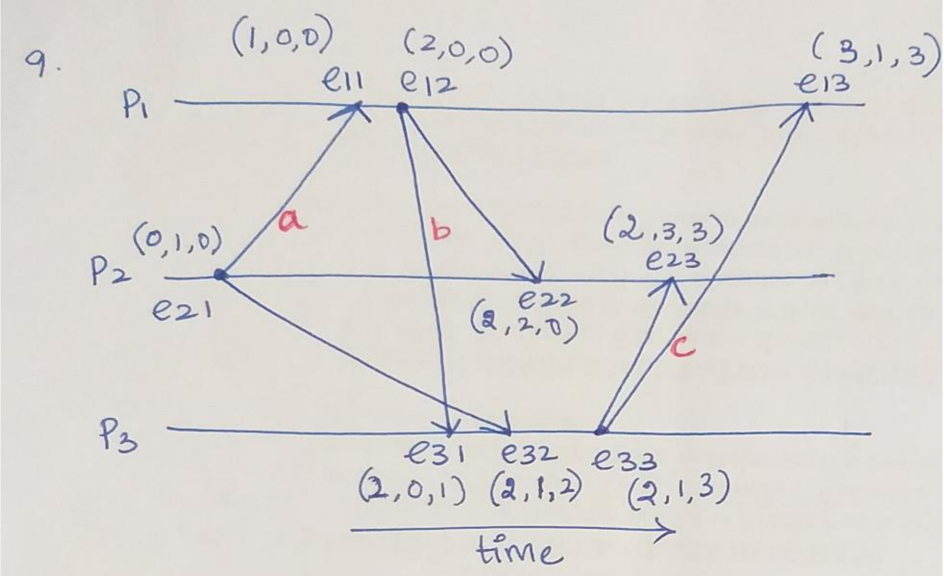
K1	1. List out various architectures for distributed systems based on Flynn's taxonomy. Single instruction stream, single data stream (SISD) Single instruction stream, multiple data stream (SIMD) Multiple instruction stream, single data stream (MISD) Multiple instruction stream, multiple data stream (MIMD)	CO1	3.1.1
K2	2. Classify the various types of transparencies which hides the implementation policies in the distributed systems Access Transparency Location Transparency Migration Transparency Relocation Transparency Replication Transparency Concurrency Transparency Failure Transparency	CO1	3.1.1
K1	3. List the three forms of load balancing Data Migration Computation Migration Distributed Scheduling	CO1	2.3.2
K2	4. Discuss and any two major issues in distributed systems Absence of Global Clock Absence of Shared Memory	CO2	2.3.2
K3	5. Given two vector timestamps '*ta*' and '*tb'*, how do you identify '*ta* < *tb'* *The clock values of event a is all less than the clock values at event b.* *C(a) < c(b). There is a happen before relation between event a and event b.* *a → b.*	CO2	1.3.1
K3	6. Given two vector timestamps '*ta*' and '*tb'*, does '*ta* $\nless$ *tb'* ***always*** mean '*ta* > *tb'* *No. It is not always true. Events a and b will be concurrent as well.*	CO2	1.3.1

Part – B (3×6 = 18 Marks)

<table>
<tr><td>K2</td><td>7. Explain synchronization and coordination mechanisms needed for distributed systems.

Synchronization/coordination mechanisms
The processes must be allowed to execute concurrently, except when they need to synchronize to exchange information, i.e., communicate about shared data. Synchronization is essential for the distributed processes to overcome the limited observation of the system state from the viewpoint of any one process. Overcoming this limited observation is necessary for taking any actions that would impact other processes. The synchronization mechanisms can also be viewed as resource management and concurrency management mechanisms to streamline the behavior of the processes that would otherwise act independently. Here are some examples of problems requiring synchronization:

• Physical clock synchronization Physical clocks ususally diverge in their values due to hardware limitations. Keeping them synchronized is a fundamental challenge to maintain common time.
• Leader election All the processes need to agree on which process will play the role of a distinguished process – called a leader process. A leader is necessary even for many distributed algorithms because there is often some asymmetry – as in initiating some action like a broadcast or collecting the state of the system, or in “regenerating” a token that gets “lost” in the system.
• Mutual exclusion This is clearly a synchronization problem because access to the critical resource(s) has to be coordinated.
• Deadlock detection and resolution Deadlock detection should be coordinated to avoid duplicate work, and deadlock resolution should be coordinated to avoid unnecessary aborts of processes.
• Termination detection This requires cooperation among the processes to detect the specific global state of quiescence.
• Garbage collection Garbage refers to objects that are no longer in use and that are not pointed to by any other process. Detecting garbage requires coordination among the processes.</td><td>CO1</td><td>2.2.2</td></tr>
<tr><td>K2</td><td>8. Outline the strategies designed for achieving the reliable and fault-tolerant distributed systems.
Refer book page no. 28 & 29
Consensus algorithms
Replication and replica management
Voting and quorum systems
Distributed databases and distributed commit
Self-stabilizing systems
Checkpointing and recovery algorithms
Failure detectors</td><td>CO1</td><td>2.2.2
2.3.2</td></tr>
<tr><td>K4</td><td>9. Consider the following diagram and analyze whether causality property is maintained or violated by computing clock values. Defend your answer. What would the clock values computed for event e13 convey?

</td><td>CO2</td><td>1.3.1
1.4.1</td></tr>
</table>

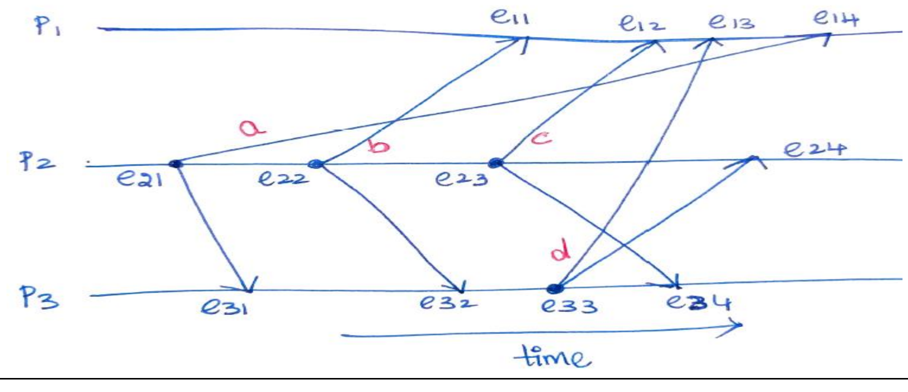
<table>
<tr>
<td></td>
<td>


P1 sends a msg to P3 (e12) after becoming aware that P2 has already communicated with P1 (e11) whereas P3's clock will not have any information about P1 when it receives (e31)
When P3 receives a message from P2 (e32), it will not carry any knowledge about P1 which will make the system outdated. hence message received at e31 should get buffered.

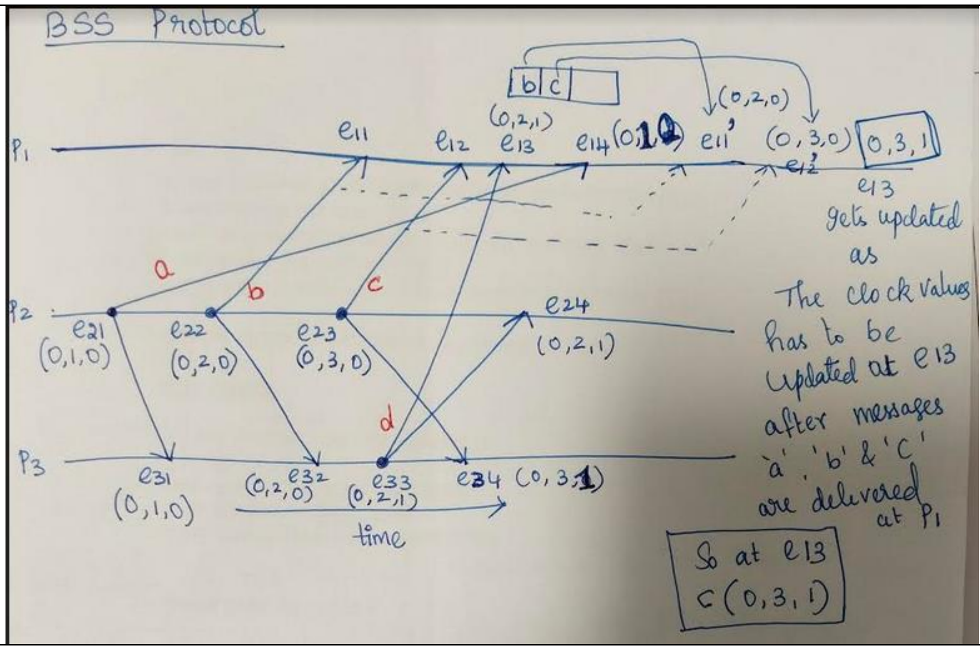
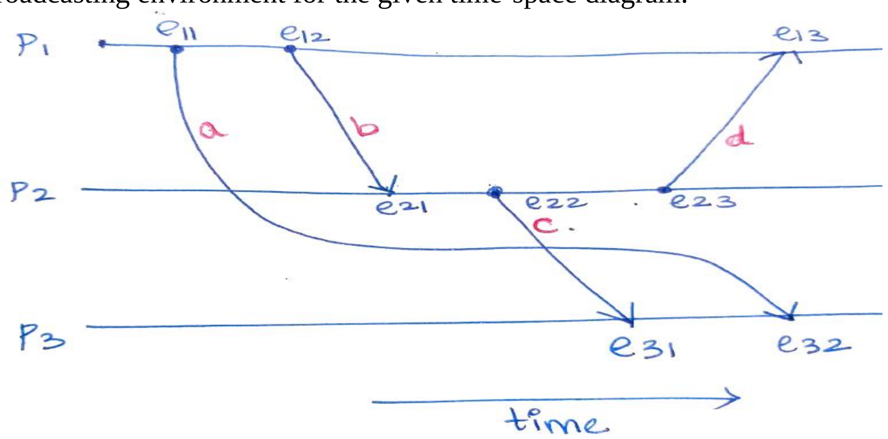
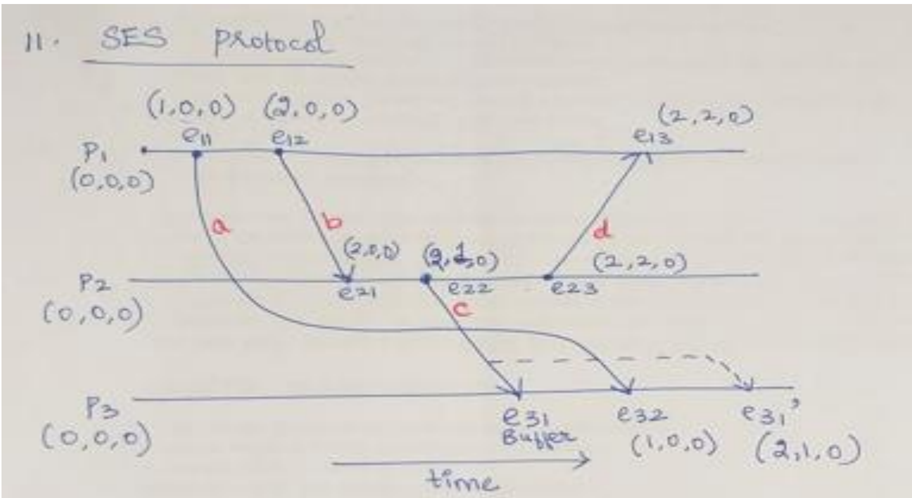
At e13 → clock values (3, 1, 3) represent 3^{rd} event
i, j, k
at P1, 1 msg received from P2 ~~from~~ (e21) &
1 msg msg received from P3 as e33 (3rd event at P3)
</td>
<td></td>
<td></td>
</tr>
</table>

Part – C (2×10 = 20 Marks)

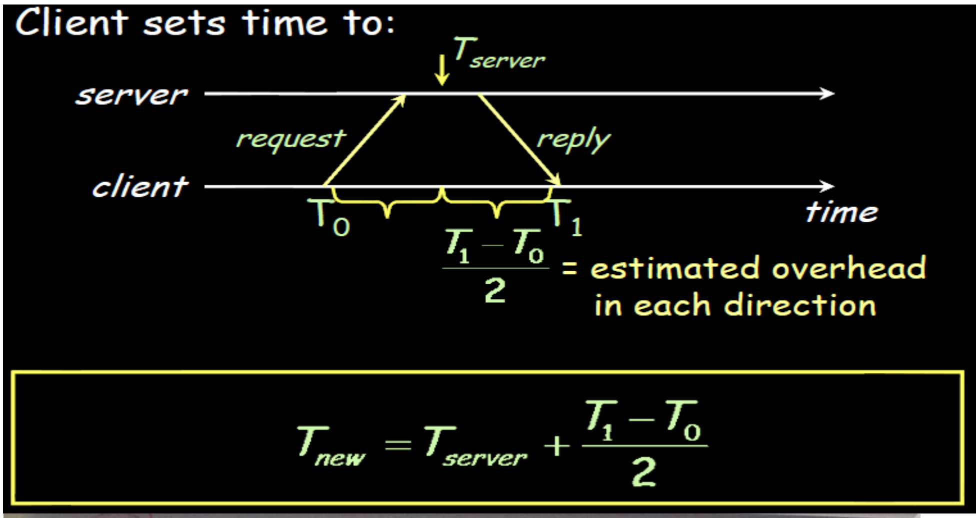
<table>
<tr>
<td>K3</td>
<td>
10. Apply suitable algorithm for the causal ordering of messages in group communication for the given time-space diagram.

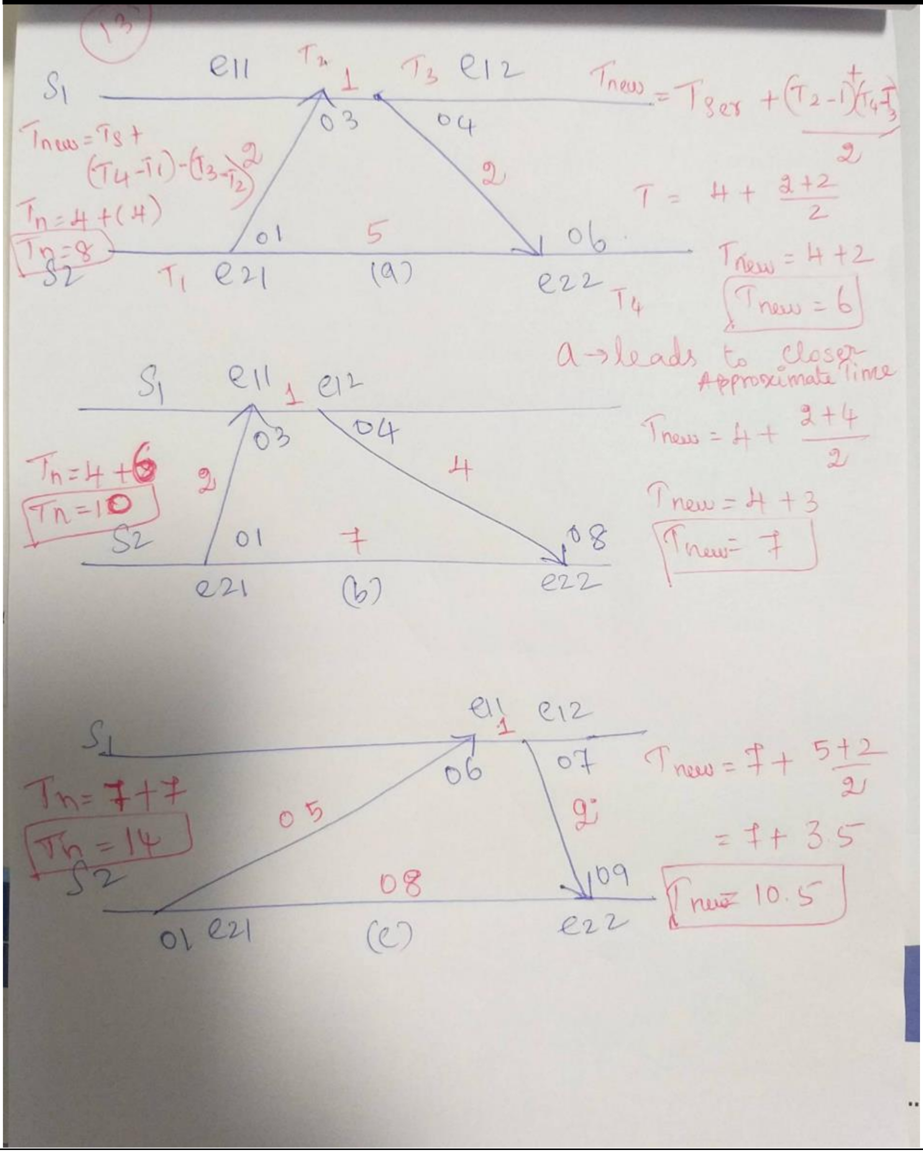


</td>
<td>CO2</td>
<td>1.3.1
1.4.1
2.3.1</td>
</tr>
</table>

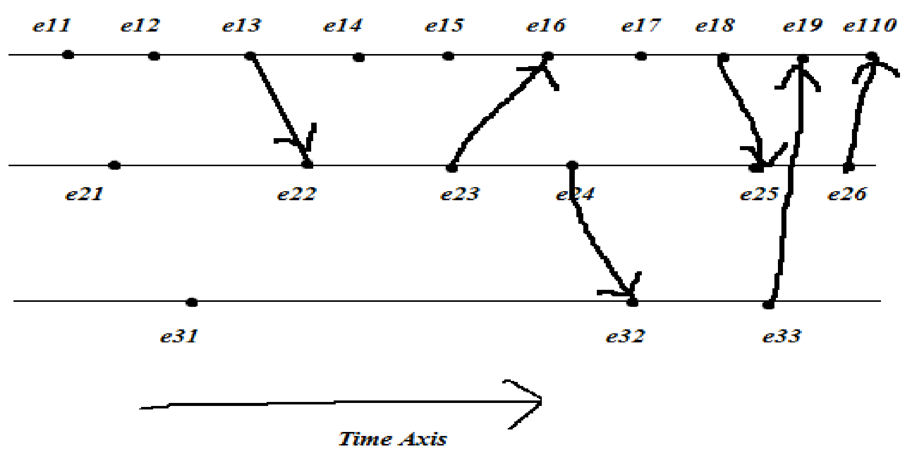
			
(OR)			
K3	11. Apply suitable algorithm for the causal ordering of messages in a non-broadcasting environment for the given time-space diagram.  	CO2	1.3.1 1.4.1 2.3.1
K3	12. Consider a suitable example and apply physical clock synchronization technique for the same. Show that physical clock synchronization is not suitable for Distributed systems through your example. Discuss the effects of clock drift and clock skew. **Ans:** Physical clocks are not suitable for Distributed systems, due to the reason that there is no global clock and clock skew and clock drift. **Clock skew**: Difference between two clocks at one point in time. **Clock Drift**: Clocks tick at different rates. Create ever-widening gap in perceived time **Christian's Algorithm**: It follows **Internal Synchronization**: Clock synchronizes to correct time by getting timings from other computers.	CO2	1.3.1 1.4.1 2.3.1

- For quartz crystal clocks, typical drift rate is about one second every 106 seconds =11.6 days
- **Quartz clock run at rate of 1.5 microseconds slower for every 35 days.**





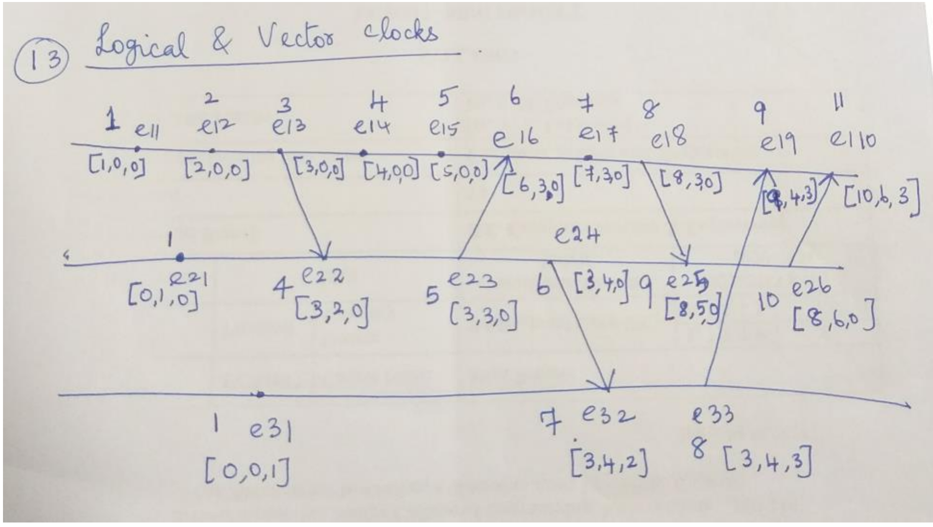
(OR)			
K3	13. Consider the following Time Space diagram and answer the following questions.	CO2	1.3.1 1.4.1 2.3.1



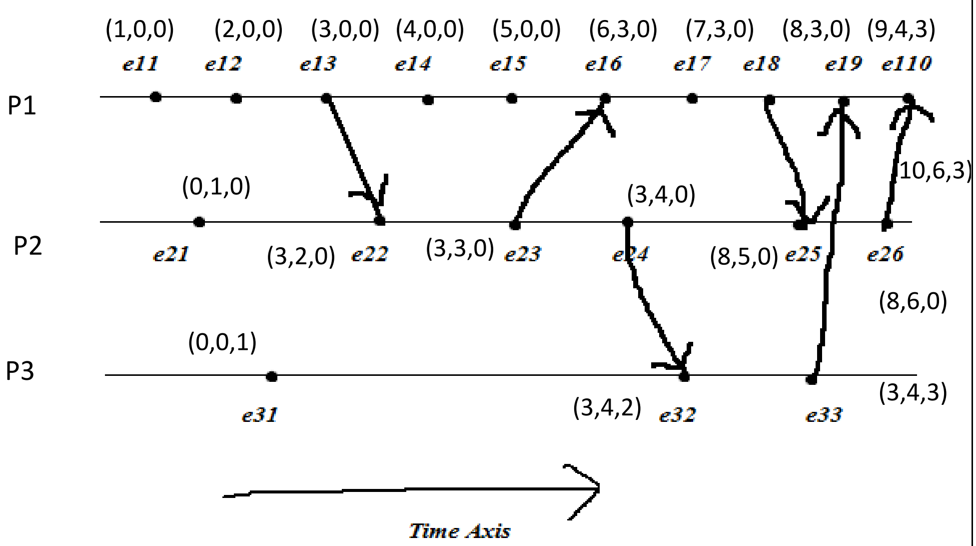
i. Apply the rules of both Lamport's Logical Clock and Vector clock and compute the values for all the events. [5]

ii. Apply the rule for causality and concurrency and show that the limitations of Lamport's clock can be resolved using vector clocks by choosing any appropriate set of events excluding e11, e12, e21 and e31. [5]

Compute logical and Vector clock values for all the events.



Vector Clocks



a) Show that the limitations of Lamport's clock can be resolved in vector clocks, with appropriate clock values of different processes

Ans: **Limitations of Lamport's Logical clock**

If a ---> b then C(a) < C(b) is true ; e13 ---> e22 and so (3,0,0) < (3,2,0) where as If C(a) < C(b) then a ---> b is not true.

	Using Vector clock this can be resolved as we update the vector of clock values for every processor in an event. Consider an event e23 in P2 with clock values as (3,3,0) and event e17 in P1 with clock values as (7,3,0) C(e23) < C(e17) then e23 ---> e17 (3,3,0) < (7,3,0) Similarly C(e25) < C(e110) then e25 ---> e110 C(e32) !< C(e19) & C(e32) !> C(e19) then e32 || e19		

-------------- ALL THE BEST --------------