Answer 2>

In Lamport's mutex algorithm paper, it is not specified anywhere as to what should be the order of sequence of the five rules. Also, the meaning of 'any' in Rule 3 and Rule 4 is not mentioned clearly, Rule 3 and Rule 4 are as follows

Rule 3: To release the resource, process Pi removes any Tm:Pi requests resource message from its request queue and sends a timestamped Pi release resource message to every other process.
Rule 4: When process Pj receives a Pi releases resource message, it removes any Tm:Pi requests resource message from its request queue.

Since the word 'any' and Tm in the above two rules leads to ambiguity among the processes decisions, hence it results in an incorrect system.
If we take 'any' to mean any request, the system might violate

(a) Safety
(b) Liveliness

Scenario:

1. Consider a case where Process 'P1' sends a request 'R_1' resource at LC = 0.
2. P2 also sends a request 'R_2' at LC = 1.
3. Now before P2 could acknowledge the request '', Process P1 sends another request 'R_1' at LC = 2

At this point of time the message queues of P1 and P2 will look like this

Case 1: Safety Violation (Removal Shown in red color)

Process P1

R_1, 0	R_2,1	R_1,2			

Queue Start

Process P2

R_1, 0	R_2,1	R_1,2			

Queue Start

4. Now, since P1 has the request, which is has the minimum LC value, P1 will get the resource and once it is done, it will remove any Tm:P1 request. Suppose it removes the request 'R_1,2' from its queue and sends the release message. On receiving this release message P2 removes the 'R_1,0'.
5. Since now, both processes will have their own requests at the top of the queue, they will both try to enter the Critical section (CS) at the same time thus violating the Safety Principle.

Case 2: Liveliness Violation (removal shown in red color)

Process P1

R_1, 0	R_2,1	R_1,2			

Queue Start

Process P2

R_1, 0	R_2,1	R_1,2			

Queue Start

4. Now, since P1 has the request which has the minimum LC value, P1 will get the resource and once it is done, it will remove any Tm:P1 request. Suppose it removes the request 'r1,0' from its queue and sends the release message. On receiving this release message P2 removes the 'r1,2'.
5. Now, Process P1 will expect P2 to enter into the Critical section and P2 will expect P1 to enter the Critical Section as it has P1's request on the top of the queue. Therefore, they will both be stuck in a deadlock thus violating the Liveliness Principle.