	Part A
1.	What is Binary Rendezvous? As said by the name, “binary” the communication between two processes are tracked and this study is called as Binary Rendezvous. In a Group Communication, there will exist a synchronous communication between any processes which are in asynchronous state at any point of time.
2.	Define check point and rollback recovery? Check Point – A Point up to which the state of the particular process is saved is called as checkpoint. Rollback Recovery – When a process is down due to some error during the execution of particular task, surely it will affect the overall outcome of the environment. Therefore, to avoid this, we have to do a process to bring back the system. This process is called as Rollback, and it is done in which that the system would have been in a consistent state.
3.	Why is rollback recovery of distributed systems complicated? Due to the existence of cascading rollback in the distributed environment. If a particular process failed, then it must be restarted (Roll Back) for bringing back to a consistent state, but as this is an distributed environment, where many process coordinated to a global task, we have to keep an eye on the other process which are involved in that coordinated activity. Even though the other system is in a good condition, it has to roll back to its previous consistent state. (i.e.) Chain Reaction
4.	Explain the concept of Centralized algorithm for total order. When process P_i wants to multicast a message M to group G: **send** $M(i, G)$ to central coordinator. When $M(i, G)$ arrives from P_i at the central coordinator: **send** $M(i, G)$ to all members of the group G. When $M(i, G)$ arrives at P_j from the central coordinator: **deliver** $M(i, G)$ to the application.
5.	Write the conditions in recording a global state How to distinguish between the messages to be recorded in the snapshot (either in a channel state or a process state) from those not to be recorded. The answer to this comes from conditions **C1** and **C2** as follows: Any message that is sent by a process before recording its snapshot, must be recorded in the global snapshot (from **C1**). Any message that is sent by a process after recording its snapshot, must not be recorded in the global snapshot (from **C2**). How to determine the instant when a process takes its snapshot. The answer to this comes from condition **C2** as follows: A process p_j must record its snapshot before processing a message m_{ij} that was sent by process p_i after recording its snapshot.
6.	List the assumptions of Binary Rendezvous? a) For the receive command, the sender must be specified. However, multiple recieve commands can exist. A type check on the data is implicitly performed. b) Send and received commands may be individually disabled or enabled. A command is disabled if it is guarded and the guard evaluates to false. The guard would likely contain an expression on some local variables. c) Synchronous communication is implemented by scheduling messages under the covers using asynchronous communication. Scheduling involves pairing of matching send and receive commands that are both enabled. The communication events for the control messages under the covers do not alter the partial order of the execution.

7.	Explain the concept of Three-phase distributed algorithm. **<u>Sender:</u>** ***Phase 1*** In the first phase, a process multicasts the message M with a locally unique tag and the local timestamp to the group members. ***Phase 2*** In the second phase, the sender process awaits a reply from all the group members who respond with a tentative proposal for a revised timestamp for that message M. The await call is non-blocking, i.e., any other messages received in the meanwhile are processed. Once all expected replies are received, the process computes the maximum of the proposed timestamps for M, and uses the maximum as the final timestamp. ***Phase 3*** In the third phase, the process multicasts the final timestamp to the group. **<u>Receivers:</u>** ***Phase 1*** In the first phase, the receiver receives the message with a tentative/proposed timestamp. It updates the variable priority that tracks the highest proposed timestamp then revises the proposed timestamp to the priority, and places the message with its tag and the revised timestamp at the tail of the queue temp_Q. In the queue, the entry is marked as undeliverable. ***Phase 2*** In the second phase, the receiver sends the revised timestamp (and the tag) back to the sender. The receiver then waits in a non-blocking manner for the final timestamp ***Phase 3*** In the third phase, the final timestamp is received from the multicaster. The corresponding message entry in temp_Q is identified using the tag (and is marked as deliverable after the revised timestamp is overwritten by the final timestamp.The queue is then resorted using the timestamp field of the entries as the key. As the queue is already sorted except for the modified entry for the message under consideration, that message entry has to be placed in its sorted position in the queue. If the message entry is at the head of the temp_Q, that entry, and all consecutive subsequent entries that are also marked as deliverable, are dequeued from temp_Q, and enqueued in deliver_Q in that order
8.	Explain the concept of Chandy–Lamport algorithm. *Marker sending rule* for process p_i (1) Process p_i records its state. (2) For each outgoing channel C on which a marker has not been sent, p_i sends a marker along C before p_i sends further messages along C. *Marker receiving rule* for process p_j On receiving a marker along channel C: **if** p_j has not recorded its state **then** Record the state of C as the empty set Execute the "marker sending rule" **else** Record the state of C as the set of messages received along C after $p_{j's}$ state was recorded and before p_j received the marker along C
9.	List the Variations of the Chandy–Lamport algorithm. a) Spezialetti–Kearns algorithm b) Venkatesan's incremental snapshot algorithm c) Helary's wave synchronization method

Part B

1. Explain the concept of checkpoint and rollback recovery and discuss the issues in failure recovery with neat diagram.

Rollback recovery treats a distributed system application to achieves fault tolerance by periodically saving the state of a process during the failure-free execution, enabling it to restart from a saved state (Checkpoint) upon a failure to reduce the amount of lost work, and this process of restarting is called as Rollback recovery.

Rollback Propagation

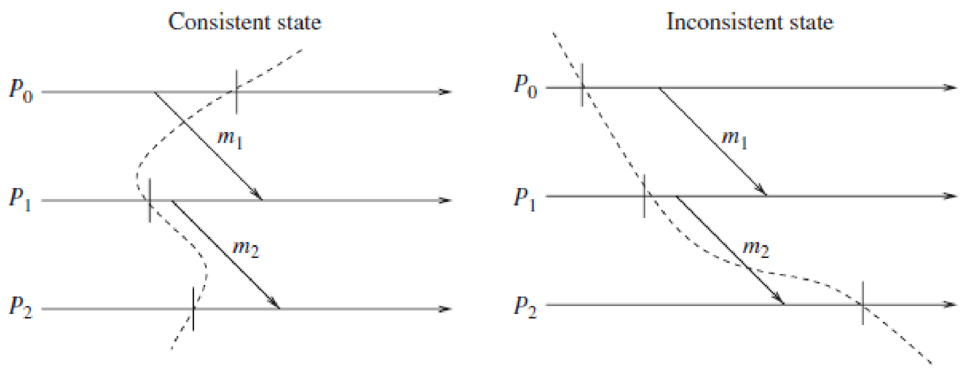
To see why rollback propagation occurs, consider the situation where the sender of a message m rolls back to a state that precedes the sending of m. The receiver of m must also roll back to a state that precedes m's receipt; otherwise, the states of the two processes would be inconsistent because they would show that message m was received without being sent, which is impossible in any correct failure-free execution. This phenomenon of cascaded rollback is called the domino effect

Note: As system must not takes it snapshot individually, this will led to a problem called as uncoordinated checkpoints. So Coordinated checkpoints has to be done, for maintain a system wide consistent state.

The approaches discussed so far implement checkpoint-based rollback recovery, which relies only on checkpoints to achieve fault-tolerance. Logbased rollback recovery combines checkpointing with logging of nondeterministic events.

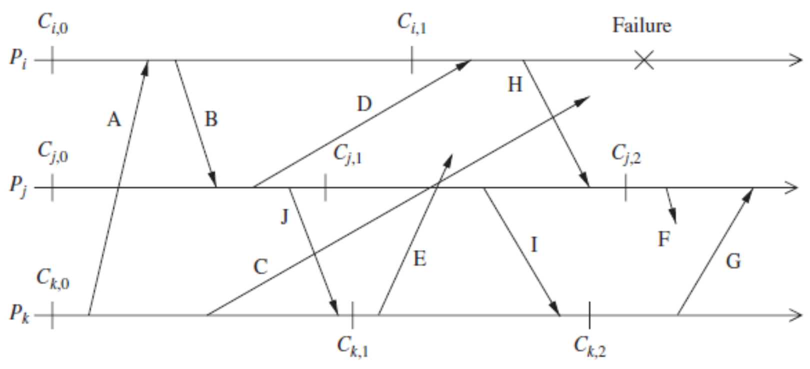
A local checkpoint is a snapshot of the state of the process at a given instance and the event of recording the state of a process is called local checkpointing.

Consistent State of Distributed Environment



A message must flow from the sender side, before taking a checkpoint, and at the other end the message must be received after the checkpoint is taken.

(i.e.) A consistent global checkpoint is a global checkpoint such that no message is sent by a process after taking its local checkpoint that is received by another process before taking its local checkpoint

	<u>Different types of messages</u> **<u>In-transit messages:</u>** A message which is sent by the sender, but haven't yet re3ceived the receiver (In transmission) **<u>Lost messages:</u>** A message which is sent by the sender is recorded and this node is fine, but on the other end the message has been received, after sometimes due to the failure of that the node, it has been restared to its previous checkpoint so this cause the message which is received to be deleted. **<u>Delayed messages:</u>** Messages whose receive is not recorded because the receiving process was either down or the message arrived after the rollback of the receiving process, are called delayed messages. **<u>Orphan messages:</u>** A message which is sent by the sender is recorded and this node is fine and on the other end the message has been received and recorded, after sometimes the sender node is failed, so it has to be restarted to its previous checkpoint thus causing the owner of that message to be lost. **<u>Duplicate messages:</u>** A message which is sent by the sender is recorded and this node is fine and on the other end the message has been received and recorded, after sometimes the sender node is failed, so it has to be restarted to its previous checkpoint thus causing this node to send the same message to the same receiver twice. Explain the types of message with your own diagram. (Mandatory) **<u>Problems in the Recovery Process when a Node gets Failed:</u>**  $C_{i,0}$ $C_{i,1}$ Failure P_i A B D H $C_{j,0}$ $C_{j,1}$ $C_{j,2}$ P_j J C E I F G $C_{k,0}$ P_k $C_{k,1}$ $C_{k,2}$ **<u>Explanation:</u>** **Refer Mail**
2.	Explain the algorithm for ordering the message using causal order. **Refer Mail**