These questions relate to reliable multicast and distributed transactions. Answers may include timelines of message transmissions/deliveries or transaction submissions/commits. For each event in the timeline, show a physical timestamp ($T_1$, $T_2$, ...), process numbers ($P_1$, $P_2$, ...), operation ('transmits', 'delivers', 'submits', 'commits'), and a numbered message ($m_1$, $m_2$, ...) or numbered transaction ($x_1$, $x_2$, ...). For example, “$T_7$: $P_1$ transmits $m_4$”. In this context define:

(a) (i) **FIFO ordering** [1 mark]
(ii) **causal ordering** [1 mark]
(iii) **total ordering** [1 mark]
(iv) **strong consistency** [1 mark]
(v) **weak consistency** [1 mark]

(b) (i) Does causal ordering imply total ordering? If so, explain why; if not, show a counterexample, labelling and explaining the violating event. [2 marks]

(ii) Does total ordering imply causal ordering? If so, explain why; if not, show a counterexample, labelling and explaining the violating event. [2 marks]

(c) A replicated database is implemented using totally ordered reliable multicast. Clients may submit transactions to any process in the group. When process $P_x$ receives a new transaction $x_i$ from a client, it will multicast the transaction to all processes, including itself. As $x_i$ is delivered by multicast, each process submits the transaction to a local ACID database. $P_x$ returns the result (abort or commit) to the client; other processes discard the transaction result.

(i) This model works well if queries do not contain the SQL **time** keyword, which is substituted with the current time when a transaction is evaluated. Explain why using **time** might be a problem and describe a solution. [4 marks]

(ii) In the first release of the database, processes submit received multicast transactions synchronously, one at a time, to the local database. In a later version, to improve performance, processes are allowed to submit multiple transactions at a time asynchronously to the local database. Why does this fail to provide strong consistency for distributed transactions? Describe a solution that might allow limited (but useful) local concurrency to be supported. [3 marks]

(iii) Describe changes to the design to support weak consistency, and describe two reasons why this might improve performance. [4 marks]