This pseudocode, executing in process \( P_j \), employs buffering to impose ordering:

\[
\text{receive(M from Pi)} \{ \quad \text{// Message M received from process Pi} \\
\quad S = \text{getSeq(M)}; \quad \text{// Extract sequence number S} \\
\quad \text{if (S == nextSeq(Sji))} \{ \quad \text{// If S is the next sequence number:} \\
\quad \quad \text{deliver(M);} \quad \text{// Deliver M to current process (Pj)} \\
\quad \quad Sji = \text{flush(HBQ, Sji);} \quad \text{// Deliver backlog from HBQ; update Sji} \\
\quad \} \quad \text{else holdback(HBQ, M);} \quad \text{// Else: Hold back M for future delivery} 
\]

(a) Explain what ordering model(s) this pseudocode implements. [2 marks]

(b) Write pseudocode (with comments) for the following functions, to be used on
the sender \( (P_i) \) or receiver \( (P_j) \), which accept \( M \) (a message), and \( S \) (a sequence number): [8 marks]

**Receiver**

- **receive_reliably(M)**: Reliably receive \( M \) from \( P_i \).
- **send_reliably(M)**: Reliably send \( M \) to \( P_j \).
- **process_ack(S)**: Handle a received ACK for \( S \) from \( P_j \).
- **timeout(S, M)**: Process a timeout for \( S \) and \( M \).

As needed, employ the following additional utility functions:

- **drop(M)**: Drop received \( M \) without delivering.
- **setSeq(M, S)**: Set sequence number \( S \) on message \( M \).
- **transmit_msg(M)**: Transmit message \( M \) to \( P_j \).
- **transmit_ack(S)**: Transmit an ACK with sequence number \( S \) to \( P_i \).
- **sched_timeout(S, M)**: Schedule timeout \( (S, M) \) to run in 5 ms.
- **cancel_timeout(S)**: If scheduled, cancel timeout for \( S \).

(c) Define the happens-before relationship. [2 marks]

(d) The pseudocode above imposes ordering on pair-wise communications. Assuming reordering but no message loss, write pseudocode (with comments) for the following functions supporting causal ordering for group communications: [8 marks]

**Receiver**

- **receive_causallyly(M)**: Causally receive from the group.

**Sender**

- **send_causallyly(M)**: Causally send to the group.

As needed, employ the following additional utility functions:

- **getVec(M)**: Retrieves the version vector from a message.
- **setVec(M, V)**: Set vector \( V \) on message \( M \).
- **testVec(LV, RV)**: Returns whether vector \( RV \) only differs from \( LV \) in that it has exactly one entry one greater than the corresponding entry in \( LV \).
- **updateVec(V)**: Returns \( V \) with the local vector entry incremented.
- **transmit_group(M)**: Transmits message \( M \) to the entire group.