DSM: Tutorial 2

Question 1

In a system with processes P1-P4:

- P1 broadcasts $[m1, m2, m3]$
- P2 broadcasts $[m4, m5]$
- $m2$ is delivered at P2 before $m5$ is broadcast.

For each of the following delivery orders at P3 and P4, the possible system types are listed:

- $[m1, m2, m3, m4, m5]$ at P3 and P4
  - Reliable, FIFO, causal and atomic.
- $[m5, m4, m3, m2, m1]$ at P3 and P4
  - Reliable.
- $[m1, m2, m3, m4, m5]$ at P3 and $[m5, m4, m3, m2, m1]$ P4
  - Reliable.
- $[m1, m4, m5, m2, m3]$ at P3 and P4
  - Reliable and FIFO.
- $[m1, m2, m3, m4, m5]$ at P3 and $[m1, m2, m3, m4]$ P4
  - Not reliable, so other types are not relevant.
- $[m1, m2, m3, m4, m5]$ at P3 and $[m1, m4, m2, m3, m5]$ P4
  - Reliable, FIFO and causal.

Question 2

Can ordering be causal but not FIFO?

No, because causal is a restriction of FIFO ordering. We assume that whenever a process broadcasts a message m it immediately delivers m to itself. Any subsequent broadcasts from that process can be said to be (possibly) causally dependent on m, and so FIFO is required for a causal ordering.

If we assume that self-deliveries may be delayed [1] then it is possible for a system to be causal but not FIFO: one process could have the delivery of its own message delayed enough for a causal relationship to be satisfied in the 'gap'.

[1] This is contrary to the model we study.

Question 3

What would have to happen in a broadcast system for it to meet agreement and integrity properties but not validity?

Agreement means that a message is delivered to everyone or no one. Integrity means that messages are delivered at most once to each process, and only if they were previously broadcast. Validity means that once broadcast, a message is eventually delivered to all correct processes. To achieve the agreement and integrity but not validity, it is necessary that a message is broadcast but delivered to no processes. This requires that the broadcasting process does not even deliver the message to itself.

Question 4

What would happen if (m, s, i) was added to $recorded$ in the `broadcast` function below? Which properties would be violated?

```
fun broadcast(m):
    for each processes p, do:
        send (m, s, i) to p
    s++
```

```
fun receive(m, s, i):
    if (recorded does not contain (m, s, i)):
        deliver(m)
        add (m, s, i) to recorded
        for each process p, do:
            send (m, s, i) to p
```

On receipt of a message, all processes would determine that it has not been seen before and deliver it correctly, apart from the process that initially broadcast it. That process will believe it has already seen that message (because it is in $recorded$), and so it will not be delivered. This violates agreement (because the message has been delivered to some but not all processes) and validity (because a broadcasted message will never be delivered at one host). Integrity is not violated.