Question 1. a, b, c are three events in a distributed system, and no two events belong to the same process. Using Lamport's definition of happened before (≺) and concurrent (||) events, comment on the truth of the following statements:

\[
\begin{align*}
\text{i} & \quad a \parallel b \parallel b \prec c \parallel a \prec c. \\
\text{ii} & \quad a \parallel b \parallel b \parallel c \parallel a \parallel c.
\end{align*}
\]

Question 2. Here is a description of Dekker’s solution, the first known two-process mutual exclusion algorithm on the shared memory model.

```
program dekker (for two processes 0 and 1)
define flag[0], flag[1] : shared boolean;
      turn: shared integer
initially flag[0] = false, flag[1] = false, turn = 0 or 1

{program for process 0}
1   do true []
2       flag[0] = true;
3       do (flag[1] []
4           if turn = 1 []
5               flag[0] := false;
6           do turn = 1 [] skip od;
7               flag[0] := true;
8          fi;
9       od;
10   critical section;
11   flag[0] = false; turn := 1;
```
12 non-critical section codes;
13 od

{program for process 1}
14 do true []
15 flag[1] = true;
16 do (flag[0] []
17 if turn = 0 []
18 flag[1] := false;
19 do turn = 0 [] skip od;
20 flag[1] := true;
21 fi;
22 od;
23 critical section;
24 flag[1] = false; turn := 0;
25 non-critical section codes;
26 od

Understand how the algorithm works. Assume both processes want to enter their CS. Then determine what is the maximum number of times a process can enter its CS before the other process is able to do so. Clearly justify your answer.

Question 3. Three philosophers 0, 1, 2 are sitting around a table. Each philosopher’s life alternates between reading and writing. There are three books on the table B0, B1, B2 – each book is placed between a pair of philosophers as shown below. While reading, a philosopher needs to grab two books – one from the right and one from the left. After a philosopher finishes reading, (s)he takes notes, and puts the books back on the table.
(a) Propose a solution by describing the life of a philosopher. A correct solution implies that each philosopher can eventually grab both books and complete the write operation infinitely often. Your solution must work with a strongly fair scheduler (but may not work with a weakly fair scheduler).

(b) Next, propose a solution (once again by describing the life of a philosopher) that will work with a weakly fair scheduler too. Provide brief arguments why your solution will work.

**Question 4.** A generalized version of the mutual exclusion problem in which up to \( L \) processes (\( L \geq 1 \)) are allowed to be in their critical sections simultaneously is known as the \( L \)-exclusion problem. Precisely, if fewer than \( L \) processes are in the critical section at any time, and one more process wants to enter its critical section, then it must be allowed to do so. Modify Ricart-Agrawala's algorithm to solve the \( L \)-exclusion problem.

**Question 5.** 13. Consider an array of \( n \) (\( n > 3 \)) processes. Starting from a terminal process, mark it as process 0, and mark the processes alternately as even and odd. Assume that the even processes have states \( \{0, 2\} \), and the odd processes have states \( \{1, 3\} \). The system uses the state-reading model of shared memory, and distributed scheduling of actions. From an unknown starting state, each process executes the following program:

```
program alternator {for process i}
    define s \( \in \{0, 1, 2, 3\} \). \{state of a process\}

    do {j: j \in neighbor(i)::s[j] = s[i]+1 mod 4 \( \in \{0, 1\} \) \& s[i] := s[i] + 2 mod 4} od
```

Determine the steady state behavior of the above system of processes. What is the maximum number of processes that can eventually execute their actions concurrently in the steady state? Show your analysis.