Second Homework of Concurrent Systems

Exercise 1. Consider the *compare_and_swap* primitive, that atomically performs the following operations:

operation $X.compare\_and\_swap(old, new)$
(1) $app \leftarrow X$
(2) **if** $(X = old)$ **then** $X \leftarrow new$
(3) **return** app

Write a *lock*/*unlock* protocol based on this primitive. Is the protocol you developed starvation free? If yes, given a formal proof; if not, give a counterexample.

Exercise 2. Consider Lamport's Bakery algorithm (Fig. 2.25 of Raynal's book) and show an execution that leads to an unbounded sequence of tickets.

Exercise 3. Consider Aravinds's bounded algorithm (Fig. 2.27 of Raynal's book together with instruction (7')). Prove that it satisfies deadlock freedom (IMPORTANT: you cannot use the fact that it satisfies bounded bypass, as we proved in class!).

Exercise 4. Consider Aravinds's bounded algorithm (Fig. 2.27 of Raynal's book together with instruction (7')). What liveness property holds if the reset of DATE happens at $n + k$ (instead of at $2n$), for $k \in \{1, \ldots, n-1\}$? Justify your answer, better if you can provide a formal proof.