Assignment 19

State the bakery algorithm. List its properties and prove each of them.

SHARED:

inter[n], number[n]

Code for process i

inter[i] = true
number[i] = max (number[1], …, number[n]) + 1
while (exist k != i s.t. : inter[k] == true AND (number[i], i) > number[k], k))
{}
<CS>
inter[i] = false

Mutual exclusion:

Assume for the sake of the contradiction that the bakery algorithm doesn't provide mutual exclusion. Assuming a set A of processes interested by enter in the CS. Assuming a $a \in A$ in the CS, then examining the code we have:
$Write_a$(inter[a] = true) → $Write_a$(number[a] = 1) →
$Read_a$(not exist k != i s.t. : inter[k] == true AND (number[a], a) > number[k], k))
We can now show that even if a process $b \in A$ is interested by enter in the CS, he will not success.
We're doing this by showing that to enter in the CS, b must write and read:
$Write_b$(inter[b] = true) → $Write_b$(number[b] = 2) →
$Read_b$(not exist k != i s.t. : inter[k] == true AND (number[b], b) > number[k], k))
which is not possible since we have a != b such that inter[a] = true AND (number[b], b) > (number[a], a)
Then, the process b have no access to the CS, contradicting the assumption.

■

Starvation freedom:

Assume for the sake of the contradiction that the bakery algorithm is not provide starvation freedom. Assuming a set A of processes interested by enter in the CS. Assuming a $a \in A$ stuck forever in the entry code (by assumption that the algorithm is not starvation free), then examining the code we have:
$Write_a$(inter[a] = true) → $Write_a$(number[a] = 1) →
$Read_a$(exist k != i s.t. : inter[k] == true AND (number[a], a) > number[k], k))

Then, we’re now focusing about the process k. Either the process k is also stuck on the while loop and so by induction we focusing on the process j which is the process which block the process k, or the process is in the CS, and when k will enter in the exit code, he will state inter[k] as false, and so, when a will have running time, he will be assure to enter in the CS, contradicting the assumption and finishing to prove the claim.