3) Algorithme de la boulangerie de Lamport (1974)

Il s'agit d'une implémentation du type abstrait VERROU qui convient pour N fils d'exécution.

```
type verrou = N le nb max de fils d'exéc.
              T un tableau de N entiers (indexé par [0..N[)
              EnCours ———————— booléens ————————
```

```
Create (n)
  v.N ← n
  v.T ← tableau indexé par [0, N[ initialisé à 0
  v.EnCours ← ——————————————— faux.
```

```
Lock (v, id)
1     v.EnCours[id] ← Vrai
2     m ← 0
3     Pour j de 0 à N-1 :
4       si v.T[j] > m
5       alors m ← v.T[j]
6     T[id] ← m+1
7     v.EnCours[id] ← Faux
8     Pour j de 0 à N-1 :
9       tant que v.EnCours[j]
10        rien
11      tant que v.T[j] ≠ 0 et (v.T[j] < v.T[id] ou (v.T[j] = v.T[id] et j < id))
12        rien
```

```
Unlock (v, id)
13    v.T[id] ← 0
```

⚠ Le test de la condition de boucle l. 11 n'est pas élémentaire.

Montrons que l'algo. de la boulangerie de Lamport garantit l'exclusion mutuelle.

Par l'absurde, supposons qu'il existe deux fils d'exécution d'identifiants k et k' qui sont tous les deux dans leur section critique à un m̃ instant t. Quitte à les échanger, on suppose que $k < k'$.

Pour $i \in [\![1,7]\!]$, on note t_i (resp t'_i) l'instant où le fil k (resp k') a exécuté la ligne i pour la dernière fois avant l'instant t.

Pour $i \in [\![9,12]\!]$, on note t_i (resp t'_i) l'instant où le fil k (resp k') a exécuté la ligne i avec $j = k'$ (resp $j = k$) pour la dernière fois avant l'instant t.

La ligne 11 n'étant pas constituée d'une instruction élémentaire, on notera au besoin $t_{11.1}$, $t_{11.2}$, $t_{11.3}$ (et $t_{11.4}$) (resp idem av t') les instants où chaque occurrence de $T[j]$ est évaluée (on suppose que les termes sont évalués de gauche à droite et m̃ que l'évalua° des et/ou est paresseuse).

De plus, on notera T au lieu de $v.T$ pour alléger le raisonnement.
On remarque que les cases $T[k]$ et $T[k']$ sont constantes entre $\max(t_6, t'_6)$ et t.

- Si $\max(t_6, t'_6) \leq \min(t_{11}, t'_{11})$ i.e. si les 2 fils ont bien fini de piocher leur tickets avant de s'interroger l'un l'autre l 11.
  alors $T[k]$ et $T[k']$ ont la m̃ valeur aux instants t_{11} et t'_{11}, et par construc° (cf l6) ces valeurs sont non nulles.
  pas besoin de préciser $t_{11.1}$ ou $t_{11.2}$ puisque $T[k]$ et $T[k']$ ne changent pas...
  Le fait que k sorte de la boucle à l'instant t_{11} implique alors $T[k'] \geq T[k]$ et $(T[k'] \neq T[k]$ ou $k' < k)$, soit, puisque $k' < k$ est faux, $T[k'] > T[k]$.
  Le fait que k' sorte de la boucle à l'instant t'_{11} implique quant à lui $T[k] \geq T[k']$ et ... , ce qui suffit à établir une contradiction.

- Sinon, on a $t_6 \geq t'_{11}$ ou $t'_6 \geq t_{11}$ (car $t_6 \geq t_{11}$ et $t'_6 \geq t'_{11}$ sont exclus puisque chaque fil exécute ses instructions séquentiellement.)
  - Si $t_6 \geq t'_{11}$, on a $\max(t_6, t'_6) = t_6$ (car $t'_6 < t'_{11}$)
    i.e si k' demande son ticket à k avant que k n'ait reçu son ticket l6.
    On a $t_6 > t'_9$, or en t'_9 EnCours$[k] = F$ et de t_1 à t_6 EnCours$[k] = V$, dc $t_1 > t'_{10}$, et donc a fortiori $t_1 > t'_6$ donc k calcule m après que Tickets$[k']$ soit fixé
    donc par construction $T[k] > T[k']$ de l'instant t_6 à l'instant t.
    Par ailleurs en $t_{11} > t_1 > t'_6$, on a $T[k'] \neq 0$, donc le fait que k sorte de la boucle assure $T[k'] \geq T[k]$, ce qui reste vrai jusqu'au temps t, d'où une contradiction !
  - Le cas $t'_6 \geq t_{11}$ est parfaitem^t symétrique (car on n'a pas utilisé ici l'hyp. $k < k'$).

Ainsi le cas où k et k' sont en section critique simultanément est impossible.