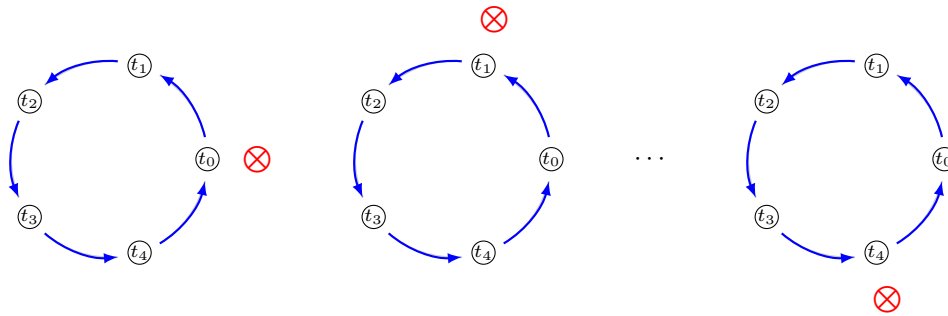


# Ring

This exercise is about handling a token sequentially using multiple threads in a ring. A token is an object, represented by the  $\otimes$  in the figure below, that has to be processed by  $T$  threads, one at a time, sequentially which means that thread 0 must process it first, then thread 1, then thread 2 and so on. This has to be repeated for as many times as a given number of loops  $L$ .



The initial, provided code only works when the number of threads  $T$  is equal to one and, therefore, consists of one simple loop over  $L$  where, at each iteration, the token is processed through the `process` function:

```
for(l=0; l<L; l++){  
    printf("Loop %2d\n",l);  
    process(&token);  
}
```

## 1 Package content

In the `ring` directory you will find the following files:


- `main.c`: this file contains the main program that first initialises the token, uses the above code to process it  $L$  times and then checks that the result is correct. **Only this file has to be modified for this exercise.**
- `aux.c`, `aux.h`: these two files contain auxiliary routines and **must not be modified.**

The code can be compiled with the `make` command: just type `make` inside the `ring` directory; this will generate a `main` program that can be run like this:

```
$ ./main L T
```

where `L` and `T` are the number of loops and threads, respectively.

## 2 Assignment

-  Extend the provided code to make it work with multiple threads  $T$ . Make sure that the processing order of the token is respected, i.e., at every loop thread  $t$  processes the token before  $t + 1$  and that all operations of loop  $l$  are finished before starting those of loop  $l + 1$ . Also, make sure that this code works for **any** number of threads  $T$ .