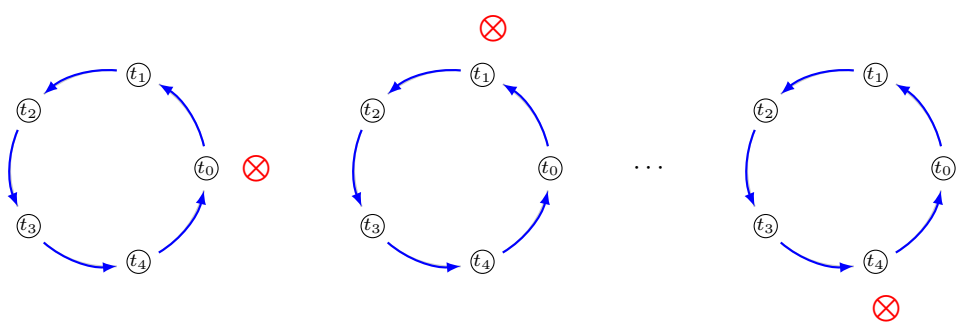
Circular pipeline

This exercise is about handling a token, represented by the ⊗ in the figure below, in a circular pipeline of S stages (or steps), for I iterations, using multiple threads. Two constraints have to be satisfied:

1. at each iteration, the stages have to be executed one at a time, sequentially which means that stage 0 must precede stage 1 which must precede stage 2 and so on.

2. There have to be as many threads as the number of stages S and only thread t can perform stage t.



The initial, provided code is

```
for(i=0; i<I; i++){
  printf("Iteration %2d\n",i);
  for(s=0; s<S; s++){
      process(&token, s);
    }
  }
}
```

and it is broken because it is not parallelized and thus it cannot satisfy constraint 2) above. At each stage of each iteration, the token is processed through the `process` function.

1 Package content

In the `circular_pipeline` 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 in I iterations with S stages each 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 `circular_pipeline` directory; this will generate a `main` program that can be run like this:

```
$ ./main I S
```

where `I` and `S` are the number of iterations and steps, respectively.

2 Assignment

- Extend the provided code to make it work with multiple threads in such a way that constraints 1) and 2) above are satisfied. Make sure that that the processing order of the token is respected, i.e., at every iteration thread s processes the token before $s+1$ and that all operations of iteration i are finished before starting those of iteration $i+1$. Also, make sure that this code works for **any** number of stages S.