

Homework 1 (due Mon. 6/6)

1. Barrier is a synchronization construct which permits a set of threads to sync up at some point in their execution, i.e. a thread will block when it reaches the barrier until all N threads reach the same point (where N is initialization parameter).

Using the primitives shown in class (or any alternative you are familiar with) implement a barrier. That is, implement

- a type `Barrier`
- routines `barrier_set` and `barrier_wait`, that operate on entities of type `Barrier`, and create and initialize the barrier, or implement the synchronization, respectively.

2. *Optional, interview-style “tricky” question, just for fun:* For regular (non-recursive) mutexes, we have used a block-structured “Lock” construct (i.e., unlocking a mutex is implicit at the end of the statement/block under the “Lock”). Can you use C macros to define a block-structured “Lock” construct from a pair of lock/unlock routines? (C++-style local variable definitions are ok, in case you need them. Alternatively, you may need to change the lock/unlock routines slightly to control what they return.)