```
********
Homework
********
```

1. (easy) Derive the version of lock framed wrt. the heap σ_s.

$$\begin{array}{l} \mathsf{lock}\ l : [h].\ \{\sigma_s = h \wedge \alpha_s = \mathbb{1} \wedge \mu_s = \cancel{\mathsf{own}}\} \\ \qquad\qquad \{\exists h'.\, \sigma_s = h' \mathbin{\dot\cup} h \wedge [h/\sigma, \alpha_o/\alpha]I \wedge \alpha_s = \mathbb{1} \wedge \mu_s = \mathsf{own}\} \end{array}$$

2. (semi-easy) Consider the program decr, which locks, decreases x, then unlocks. Can you verify that program with the PCM of nats? Can you think of PCMs in which it can be verified?

   - We either have to switch to integers instead of nats, or to histories PCM.
   - We can also keep a PCM of pairs (a, b). The component a says how much we've incremented, b says how much we've decremented. But that's basically an encoding of integers.

3. (harder) Generalize the implementation of incr, so that it takes an argument $k : \mathsf{nat}$, and increments x by k.

   Then prove that the program that iterates over the list $[k_1, k_2, \ldots, k_n]$, and forks the thread that incr's over each element, upon termination, increments x by $k_1 + \cdots + k_n$.

$$\begin{array}{rcl} \mathsf{iterate\ nil} & = & \mathsf{return}() \\ \mathsf{iterate}\ (x :: xs) & = & \mathsf{incr}(x) \parallel \mathsf{iterate}\ xs;\ \mathsf{return}() \end{array}$$