Separation Logic exercises, lecture 3

Exercise 1. Use rule Iter-past to prove `length`: $I\,k = \ldots$

$$\frac{\forall \quad \{\quad\}\{\quad\}\ \texttt{incr r}\ \{\quad\}\{\quad\}}{\{\quad\}\ \texttt{let r = ref 0 in}\ \{\quad\}\ \{\quad\}\ \texttt{iter (fun x -> incr r)}\ l\ \{\quad\}\ \{\quad\}\ \texttt{!r}\ \{\lambda v.\ulcorner v = \quad\urcorner\}}$$

Exercise 2. Use rule Iter-future to prove `length`: $I'\,k = \ldots$

Exercise 3. Give an example where Iter-Past is insufficient. Adapt the premise of the rule to allow your example.

Exercise 4. Give heaps satisfying the following predicates:

(1). $\ulcorner\urcorner \mathrel{-\!\!*} (1 \mapsto 2)$: . . . . . (2). $\ulcorner\mathsf{False}\urcorner \mathrel{-\!\!*} (1 \mapsto 2)$: . . . . . .

(3). $\ulcorner x \geqslant 1\urcorner \mathrel{-\!\!*} \ulcorner x \geqslant 0\urcorner$: . . . . . (4). $(1 \mapsto 4) \mathrel{-\!\!*} (1 \mapsto 4) * (2 \mapsto 3)$: . . . . . .

(5). $(1 \mapsto 2) \mathrel{-\!\!*} (1 \mapsto 2)$: . . . . . (6). $(1 \mapsto 2) \mathrel{-\!\!*} \ulcorner\mathsf{False}\urcorner$: . . . . . .

(7). $(1 \mapsto 2) \mathrel{-\!\!*} \ulcorner\urcorner$: . . . . . (8). $(1 \mapsto 2) \mathrel{-\!\!*} (1 \mapsto 3)$: . . . . . .

Exercise 5. Among the following heap entailments, which hold?

(1). $P \rhd (Q \mathrel{-\!\!*} P * Q)$ (2). $(Q \mathrel{-\!\!*} P * Q) \rhd P$ (3). $(1 \mapsto 2) \mathrel{-\!\!*} (1 \mapsto 3) \rhd \ulcorner\mathsf{False}\urcorner$

(4). $(1 \mapsto 2) \mathrel{-\!\!*} (1 \mapsto 2 * 2 \mapsto 8) \rhd 2 \mapsto 8$ (5). $\ulcorner\urcorner \mathrel{-\!\!*} P \rhd P$ (6). $P \rhd \ulcorner\urcorner \mathrel{-\!\!*} P$

(7). $\ulcorner\urcorner \rhd (P \mathrel{-\!\!*} Q \mathrel{-\!\!*} P * Q)$ (8). $\ulcorner P \rhd Q\urcorner \rhd (P \mathrel{-\!\!*} Q)$ (9). $(P \mathrel{-\!\!*} Q) \rhd \ulcorner P \rhd Q\urcorner$

Exercise 6. What is a correct postcondition for `p.hd`?

Exercise 7. Give an example program for which this postcondition is insufficient, for example for list of mutable objects. What is a postcondition for `p.hd` that fixes this problem?

Exercise 8. specify the function `miter`, using an invariant of the form $J\,K\,K'$, where K is remaining to process and K' is a result list.

Exercise 9. States before/after declarations; specify and prove the function.

```
{                          }
let r = ref 0
{                          }
let s = ref n
{                          }
{                          }
let p = newlock ()
{                          }
```

```
let concurrent_step () =
  {                          }
  acquire p;
  {                          } ...
  {                          }
  incr r; decr s;
  {                          }
  {                          }
  release p
  {                          }
```

Exercise 10. Find a lock invariant and then prove {True} program {True}. Hint: you can define a (temporary!) notation $\ell \hookrightarrow v \;\equiv\; \ell \overset{1/2}{\mapsto} v$.

```
                    {⌜True⌝}
        let r = ref 0 and r1 = ref 0 and r2 = ref 0
              {                    }
          {                              }
                    let p = newlock ()
              {                    }
        {                                    }
```

`{ }`	`{ }`
`acquire p;`	`acquire p;`
`{ } . . . .`	`{ } . . . .`
`{ }...`	`{ }...`
`{ }`	`{ }`
`r := !r + 1;`	`r := !r + 1;`
`{ }`	`{ }`
`{ }`	`{ }`
`r1 := !r1 + 1;`	`r2 := !r2 + 1;`
`{ }`	`{ }`
`{ }`	`{ }`
`release p;`	`release p;`
`{ }`	`{ }`

```
              {                    }
                    acquire p;
    {                                    }...
              {                    }
                    assert (!r == 2);
                    {⌜True⌝}
```