Exercise 1. What problem is there, if $x \leadsto R X$ is $x \mapsto X$ (i.e. $R = \mathsf{Ref}$)?

Exercise 2. How to generalize the specification to solve this problem?

Exercise 3. specify functions over queues using a higher-order representation predicate written $p \leadsto \mathsf{Queueof}\, R\, L$.
Shorthand: just write "$\mathsf{Q}\, R$" instead of "$\mathsf{Queueof}\, R$".

$$\{ \qquad\qquad\qquad \}\ (\texttt{create()})\ \{ \qquad\qquad\qquad \}$$

$$\{ \qquad\qquad\qquad \}\ (\texttt{push}\, x\, p)\ \{ \qquad\qquad\qquad \}$$

$$\{ \qquad\qquad\qquad \}\ (\texttt{pop}\, p)\ \{ \qquad\qquad\qquad \}$$

$$\{ \qquad\qquad\qquad \}\ (\texttt{concat}\, p\, p')\ \{ \qquad\qquad\qquad \}$$

Exercise 4. specify a function $\texttt{copy}\, f\, p$ that duplicables a mutable queue specified using $\mathsf{Queueof}$, where f is a function to duplicate items.

$$\begin{array}{l} \big(\forall x X.\ \{ \qquad\qquad \}\ (f\ x)\ \{ \qquad\qquad \}\big) \\ \Rightarrow\ \{ \qquad\qquad \}\ (\texttt{copy}\ f\ p)\ \{ \qquad\qquad \} \end{array}$$

Exercise 5. rewrite the specification of $\mathsf{Mlistof}$ using $\mathsf{MCellof}$.

$$p \leadsto \mathsf{MCellof}\, R_1\, V_1\, R_2\, V_2 \ \equiv\ \exists v_1 v_2.\quad \begin{array}{l} p \leadsto \{|\mathrm{hd}{=}v_1;\ \mathrm{tl}{=}v_2|\} \\ \star\ v_1 \leadsto R_1\, V_1 \\ \star\ v_2 \leadsto R_2\, V_2 \end{array}$$

$$p \leadsto \mathsf{Mlistof}\, R\, L\ \equiv\ \begin{array}{l} \mathsf{match}\ L\ \mathsf{with} \\ \mid \mathsf{nil}\ \Rightarrow\ \ulcorner p = \mathsf{null} \urcorner \\ \mid X :: L'\ \Rightarrow \end{array}$$

Exercise 6. rewrite the specification of $\mathsf{Narytreeof}$ using Nodeof.

$$\begin{array}{l} p \leadsto \mathsf{Narytreeof}\, R\, T \equiv \\ \quad \mathsf{match}\ T\ \mathsf{with} \\ \quad \mid \mathsf{Leaf}\ \Rightarrow\ \ulcorner p = \mathsf{null} \urcorner \\ \quad \mid \mathsf{Node}\ X\ L\ \Rightarrow \end{array}$$

Exercise 7. specify the function `miter`, using an invariant of the form $J\,K\,K'$, describing the state before and the state after the iteration.

$$\forall f p R L J.\quad \big(\forall x X \qquad\qquad . \; \{x \rightsquigarrow R\,X \qquad\qquad\} \; (f\,x) \; \{\lambda\_. \qquad\qquad\}\big)$$

$$\Rightarrow \{p \rightsquigarrow \mathsf{Mlistof}\,R\,L \star \qquad\qquad\} \; (\mathtt{miter}\,f\,p) \; \{\lambda\_. \qquad\qquad\}$$

Exercise 8. using the representation predicates Ref (i.e. $x \rightsquigarrow \mathsf{Ref}\,X \equiv x \mapsto X$) and Mlistof, specify the function `(fun x -> incr x)` and `incr_all`. What is $J\ K\ K'$?

```
let incr_all p = miter (fun x -> incr x) p

let example_p = { hd = ref 5; tl = { hd = ref 3; tl = null } }
```

$$\{ \qquad\qquad\qquad \} \; (\mathtt{incr\ x}) \; \{\lambda\_. \qquad\qquad\qquad \}$$

$$\{ \qquad\qquad\qquad \} \; (\mathtt{incr\_all}\,p) \; \{\lambda\_. \qquad\qquad\qquad \}$$

$J\ K\ K' =$

Exercise 9. Describe the state before each instruction (except line 5). Explicit the instantiation of the existential in the invariant.

```
1   let r = ref 0
2   let s = ref n
3   let p = create_lock ()
4
5   let concurrent_step () =
6      acquire_lock p;
7      incr r;
8      decr s;
9      release_lock p
```

Exercise 10. state a conversion rule relating $p \rightsquigarrow \mathsf{Cellsof}\, R\, M$ with a predicate of the form $p \rightsquigarrow \mathsf{Cellsof}\,\mathsf{Id}\, M'$.
Hint: $(R : A \to a \to \mathsf{Hprop})$ and $(M : \mathsf{map\,int}\, A)$ and $(M' : \mathsf{map\,int}\, a)$.

$$p \rightsquigarrow \mathsf{Cellsof}\, R\, M \quad =$$

Exercise 11. Let program be:

```
let r = ref 0
let r1 = ref 0
let r2 = ref 0
let p = create_lock()
acquire_lock p;     ||   acquire_lock p;
r := !r + 1;        ||   r := !r + 1;
r1 := !r1 + 1;      ||   r2 := !r2 + 1;
release_lock p;     ||   release_lock p;
acquire_lock p;
assert (!r == 2);
```

Give a lock invariant that allows proving {True} program {True}, then prove the triple.

Exercise 12. WP rules for load, store, alloc

$$\forall \ell v \Phi \quad \ldots\ldots\ldots\ldots \;-\!\!* \;(\ldots\ldots\ldots\ldots\ldots\ldots\ldots) \;-\!\!*\; \mathsf{wp}\,(!\ell)\,\Phi$$
$$\forall \ell v v' \Phi \quad \ldots\ldots\ldots\ldots \;-\!\!* \;(\ldots\ldots\ldots\ldots\ldots\ldots\ldots) \;-\!\!*\; \mathsf{wp}\,(\ell \leftarrow v)\,\Phi$$
$$\forall v \Phi \quad \ldots\ldots\ldots\ldots \;-\!\!* \;(\ldots\ldots\ldots\ldots\ldots\ldots\ldots) \;-\!\!*\; \mathsf{wp}\,(\mathtt{ref}\ v)\,\Phi$$