Exercise 1. Specify functions on queues using an HO representation predicate written $p \rightsquigarrow \mathsf{Queueof}\, R\, L$ (you can write "$\mathsf{Q}\, R$" instead of "$\mathsf{Queueof}\, R$").

$\forall \ldots \quad \{ \quad \} \, (\texttt{create()}) \, \{ \quad \}$

$\forall \ldots \quad \{ \quad \} \, (\texttt{push}\, x\, p) \, \{ \quad \}$

$\forall \ldots \quad \{ \quad \} \, (\texttt{pop}\, p) \, \{ \quad \}$

$\forall \ldots \quad \{ \quad \} \, (\texttt{concat}\, p\, p') \, \{ \quad \}$

Exercise 2. specify a function $\texttt{copy}\, f\, p$ that duplicates a mutable queue specified using $\mathsf{Queueof}$, where f is a function to duplicate items.

$\forall \ldots \quad (\forall x X. \, \{ \quad \} \, (f\, x) \, \{ \quad \})$

$\Rightarrow \{ \quad \} \, (\texttt{copy}\, f\, p) \, \{ \quad \}$

Exercise 3. rewrite the definition of $\mathsf{Mlistof}$ using $\mathsf{MCellof}$.

$$p \rightsquigarrow \mathsf{Mlistof}\, R\, L \equiv \mathsf{match}\, L\, \mathsf{with} \mid \mathsf{nil} \Rightarrow \ulcorner p = \mathsf{null} \urcorner \mid X :: L' \Rightarrow \ldots$$

Exercise 4. rewrite the definition of $\mathsf{Narytreeof}$ using Nodeof'.

$$p \rightsquigarrow \mathsf{Narytreeof}\, R\, T \equiv \mathsf{match}\, T\, \mathsf{with} \mid \mathsf{Leaf} \Rightarrow \ulcorner p = \mathsf{null} \urcorner \mid \mathsf{Node}\, X\, L \Rightarrow \ldots$$

Exercise 5. specify the function `miter`, using an invariant of the form $J\, K\, K'$, describing the state before and the state after the iteration.

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

$\Rightarrow \{p \rightsquigarrow \mathsf{Mlistof}\, R\, L \; * \quad \} \, (\texttt{miter}\, f\, p) \, \{\lambda\_. \quad \}$

Exercise 6. using the representation predicates Ref (i.e. $x \rightsquigarrow \mathsf{Ref}\, X \equiv x \mapsto X$) and $\mathsf{Mlistof}$, specify the function `incr` and `miter incr`. What is $J\ K\ K'$?

$\{ \quad \} \, (\texttt{incr}\ x) \, \{\lambda\_. \quad \}$

$\{ \quad \} \, (\texttt{miter incr}\, p) \, \{\lambda\_. \quad \}$

$J\ K\ K' =$

Exercise 7. Give a proof sketch of the declarations; specify and prove the function.

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

```
let concurrent_step () =
   {                              }
   acquire_lock p;
   {                              }
   {                              }
   incr r; decr s;
   {                              }
   {                              }
   release_lock p
   {                              }
```

Exercise 8. Give a lock invariant: $H =$
that allows proving the program below. Then prove the program below.

$\{\ulcorner\mathsf{True}\urcorner\}$

```
let r = ref 0 and r1 = ref 0 and 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);
```

$\{\ulcorner\mathsf{True}\urcorner\}$