Exercise:

1. specify: `let r = ref 5 and s = ref 3 and t = r`.
2. specify the state after subsequently executing: `incr r`.
3. specify the state after subsequently executing: `incr t`.

1.
2.
3.

In-place list reversal. Before the loop: After the loop: Loop invariant:

Mlength with a while loop. Before the loop:

After the loop:

where L denotes the list of items in the list segment from p (inclusive) to q (exclusive).

Loop invariant: **Exercise:** generalize MList to define $p \rightsquigarrow \mathsf{MlistSeg}\, q\, L$,

$p \rightsquigarrow \mathsf{MlistSeg}\, q\, L \quad \equiv$

Enter:

Exit:

Step:

Exercise: define the representation predicate $p \rightsquigarrow \mathsf{Queue}\, L$. **Exercise:**

define $p \rightsquigarrow \mathsf{Mtree}\, T$. **Exercise:** define $p \rightsquigarrow \mathsf{MtreeDepth}\, n\, T$ by gener-

alizing $p \rightsquigarrow \mathsf{Mtree}\, T$. **Exercise:** give an alternative definition of "$p \rightsquigarrow$

$\mathsf{MtreeDepth}\, n\, T$", this time by reusing the definition of $p \rightsquigarrow \mathsf{Mtree}\, T$ without modification. **Exercise:** define a predicate $p \rightsquigarrow \mathsf{MtreeComplete}\, T$

for describing a mutable complete binary tree, of some unspecified depth. **Exercise:** define a predicate $p \rightsquigarrow \mathsf{MsearchTree}\, E$ for describing a mutable

binary search tree storing the set of elements E. **Exercise:** specify the primitive operations on references.

`(ref v)`

`(!r)`

`(r := v)`

Give specifications for:

`(Array.get i p)`

`(Array.set i p v)`

`(Array.length p)`

`(Array.create n v)`

Interpretation of triples (1/3).
How is a triple $\{H\}\ t\ \{Q\}$ intepreted?

$$\forall m.\quad H\,m \quad\Rightarrow\quad \exists v.\,\exists m'.\ \ \langle t, m\rangle \Downarrow \langle v, m'\rangle\ \ \wedge$$

Interpretation of triples (2/3).
In Separation Logic, a triple describes only a part m_1 of the heap.
The rest of the heap, call it m_2, is assumed to remain unchanged. How is a triple $\{H\}\ t\ \{Q\}$ intepreted? What is the *natural* specification of function

`myref`? What is missing from our current interpretation of triple?