Exercise 1.  give heaps satisfying the following heap predicates

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>'1'</td>
<td>'0 = 1'</td>
</tr>
<tr>
<td>'1 = 1'</td>
<td>'1 = 1' * '0 = 1'</td>
</tr>
<tr>
<td>1 \rightarrow 2</td>
<td>(1 \rightarrow 2) * '1 = 1'</td>
</tr>
<tr>
<td>(1 \rightarrow 2) * (1 \rightarrow 3)</td>
<td>(1 \rightarrow 2) * (2 \rightarrow 1)</td>
</tr>
</tbody>
</table>

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

Exercise 3.  give heaps satisfying the following heap predicates

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>\exists x. '(1 \rightarrow x)'</td>
<td>\exists x. (1 \rightarrow x) * (2 \rightarrow x)</td>
</tr>
<tr>
<td>\exists x. x = x + 1'</td>
<td>\exists x. (x \rightarrow x + 1) * (x + 1 \rightarrow x)</td>
</tr>
<tr>
<td>\exists x. 1 \rightarrow x</td>
<td>\exists x. (x \rightarrow 1) * (x \rightarrow 2)</td>
</tr>
<tr>
<td>\exists P. 'P'</td>
<td>\exists H. H</td>
</tr>
</tbody>
</table>

Exercise 4.  in-place list reversal
State before the loop:
State after the loop:
Loop invariant:

Exercise 5.  length of mutable list using a while loop
State before the loop:
State after the loop:
Picture describing the state during the loop:

Try to state a loop invariant. What do you need?

Exercise 6.  generalize MList to define \( p \leadsto \text{MlistSeg} q L \), where \( L \) denotes the list of items in the list segment from \( p \) (inclusive) to \( q \) (exclusive):

\( p \leadsto \text{MlistSeg} q L \) \( \equiv \)
Exercise 7. length of mutable list using a while loop and MlistSeg
Loop invariant: \( \exists q, L_1, L_2. \ldots \)

Instantiate \( q, L_1, L_2 \) before the loop:
Instantiate \( q, L_1, L_2 \) after the loop:

Exercise 8. define the representation predicate \( p \leadsto \text{Queue}\ L \).

Exercise 9. define the representation predicate \( p \leadsto \text{Mtree}\ T \).

Exercise 10. define \( p \leadsto \text{MtreeDepth}\ n\ T \) by generalizing \( p \leadsto \text{Mtree}\ T \).

Exercise 11. give an alternative definition of “\( p \leadsto \text{MtreeDepth}\ n\ T \)”, this time by reusing the definition of \( p \leadsto \text{Mtree}\ T \) without modification.

Exercise 12. define a predicate \( p \leadsto \text{MtreeComplete}\ T \) for describing a mutable complete binary tree, of some unspecified depth.

Exercise 13. define a predicate \( p \leadsto \text{MsearchTree}\ E \) for describing a mutable binary search tree storing the set of elements \( E \).

Exercise 14. specify the primitive operations on references.
\[
\begin{align*}
\text{(ref } v) \\
\text{(!r)} \\
\text{(r := v)}
\end{align*}
\]

Exercise 15. Give specifications for:
\[
\begin{align*}
\text{(Array.get } i \text{ p)} \\
\text{(Array.set } i \text{ p v)} \\
\text{(Array.length } p) \\
\text{(Array.create } n \text{ v)}
\end{align*}
\]

Exercise 16. What is the natural specification of function myref? What is missing from our current interpretation of triple?