Exercise: describe the frame process for in-place increment. **Exercise:**

specify the tree copy function. **Exercise:** describe the frame process for

tree copy. **Exercise:** give small footprint specifications for array operations.

How to derive the large footprint specifications from them?

{ }	`(Array.get i p)`	{ }	
{ }	`(Array.set i p v)`	{ }	
{ }	`(Array.length p)`	{ }	

Exercise: give a small-footprint specification for quicksort. For

each of the heap implications below, state whether it is true or false.

1. $(r \mapsto 3) \star (s \mapsto 4) \rhd (s \mapsto 4) \star (r \mapsto 3)$
2. $(r \mapsto 3) \rhd (s \mapsto 4) \star (r \mapsto 3)$
3. $(s \mapsto 4) \star (r \mapsto 3) \rhd (r \mapsto 4)$
4. $(s \mapsto 4) \star (r \mapsto 3) \rhd (r \mapsto 3)$
5. $[\mathsf{False}] \star (r \mapsto 3) \rhd (s \mapsto 4) \star (r \mapsto 4)$
6. $(r \mapsto 4) \star (s \mapsto 3) \rhd [\mathsf{False}]$
7. $(r \mapsto 4) \star (r \mapsto 3) \rhd [\mathsf{False}]$
8. $(r \mapsto 3) \star (r \mapsto 3) \rhd [\mathsf{False}]$

For each of the heap implications below, state whether it is true or false.

1. $(r \mapsto 3) \rhd \exists n.\, (r \mapsto n)$
2. $\exists n.\, (r \mapsto n) \rhd (r \mapsto 3)$
3. $\exists n.\, (r \mapsto n) \star [n > 0] \rhd \exists n.\, [n > 1] \star (r \mapsto (n-1))$
4. $(r \mapsto 3) \star (s \mapsto 3) \rhd \exists n.\, (r \mapsto n) \star (s \mapsto n)$
5. $\exists n.\, (r \mapsto n) \star [n > 0] \star [n < 0] \rhd (r \mapsto n) \star (r \mapsto n)$

Exercise: show that GC-PRE is derivable from GC-POST and FRAME.

$$\frac{\{H\}\ t\ \{Q\}}{\{H \star \mathsf{GC}\}\ t\ \{Q\}}$$

Exercise: give a specification of copy in terms of MtreeComplete; which rules are used to derive this specification? **Exercise:** complete the rule

for sequences.

$$\frac{\{\qquad\}\ t_1\ \{\qquad\} \qquad \{\qquad\}\ t_2\ \{\qquad\}}{\{H\}\ (t_1\,;\,t_2)\ \{Q\}}$$

Exercise: complete the reasoning rule for let-bindings.

$$\frac{\{\qquad\}\ t_1\ \{\qquad\} \qquad \forall x.\ \{\qquad\}\ t_2\ \{\qquad\}}{\{H\}\ (\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)\ \{Q\}}$$

Exercise: instantiate the rule for let-bindings on the following code.

$$\{r \mapsto 3\}\ (\texttt{let a = !r in a+1})\ \{Q\}$$

Solution:

$$\begin{aligned} H &\equiv \\ Q &\equiv \\ Q' &\equiv \end{aligned}$$

Reasoning rule for values:

$$\frac{\rhd}{\{H\}\ v\ \{Q\}}$$

Exercise: specify the interface for mutable queues in terms of:

$$p \rightsquigarrow \mathsf{Queue}\, L$$

Remark: items are pushed to the back of list, and popped from the head; transfer migrates items from the second queue to the back of the first.

(`create ()`)

(`is_empty q`)

(`push x q`)

(`pop q`)

(`transfer q1 q2`)