

SEPARATION LOGIC EXERCISES, LECTURE 1

Exercise 1. give heaps satisfying the following heap predicates

\top		$\top = \top$	
$\top = \top$		$\top = \top * \top = \top$	
$1 \mapsto 2$		$(1 \mapsto 2) * \top = \top$	
$(1 \mapsto 2) * (1 \mapsto 3)$		$(1 \mapsto 2) * (2 \mapsto 1)$	

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

$\exists x. \top(1 \mapsto x)$		$\exists x. (1 \mapsto x) * (2 \mapsto x)$	
$\exists x. \top x = x + 1$		$\exists x. (x \mapsto x + 1) * (x + 1 \mapsto x)$	
$\exists x. 1 \mapsto x$		$\exists x. (x \mapsto 1) * (x \mapsto 2)$	
$\exists P. \top P$		$\exists H. H$	

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 $p \rightsquigarrow \text{MlistSeg } q L$, where L denotes the list of items in the list segment from p (inclusive) to q (exclusive):

$p \rightsquigarrow \text{MlistSeg } q L \equiv$

Exercise 7. length of mutable list using a while loop and MlistSeg

Loop invariant: $\exists q, L_1, L_2. \dots$

Instantiate q, L_1, L_2 before the loop:

Instantiate q, L_1, L_2 after the loop:

Exercise 8. define the representation predicate $p \rightsquigarrow \text{Queue } L$.

Exercise 9. define the representation predicate $p \rightsquigarrow \text{Mtree } T$.

Exercise 10. define $p \rightsquigarrow \text{MtreeDepth } n T$ by generalizing $p \rightsquigarrow \text{Mtree } T$.

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

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

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

Exercise 14. specify the primitive operations on references.

`(ref v)`

`(!r)`

`(r := v)`

Exercise 15. Give specifications for:

`(Array.get i p)`

`(Array.set i p v)`

`(Array.length p)`

`(Array.create n v)`

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