Bhavesh Rajesh Talreja (A20516822)
CS 536 HW7

1. Linked List and Separation logic

Task 1.1

Given:

Input: List *L* with length m+3, start of this list pointed by *head*
Output: List L_1 with length 3, start of this list pointed by $head_1$ and List L_2 with length m, start of this list pointed by $head_2$

Ans.

S ≜ $head_1 := head;$
$x_0 := !\,(head_1 + 1);$
$x_1 := !\,(x_0 + 1);$
$head_2 := !\,(x_1 + 1);$
$!\,(x_1 + 1) := nil$

By this program, after the 3rd element, in List L, we break the connection to the remaining part of the list by changing the pointer to point to nil. There by achieving our goal to create two lists from a list as per the specification.

Task 1.2

Given Program:

S ≜ $head_1 := head;$
$x_0 := !\,(head_1 + 1);$
$x_1 := !\,(x_0 + 1);$
$head_2 := !\,(x_1 + 1);$
$!\,(x_1 + 1) := nil$

Ans.

Here we must provide a precondition, a postcondition and attempt a full proof outline.

By referencing the example from the class:

The precondition that worked for me: {list(*L*, *head*, m+3)}
The postcondition that worked for me: {list(L_1, $head_1$, 3) * list(L_2, $head_2$, m)}

The list(L, i, n) predicate allows us to re-write the postcondition as follows:
{list(L_1, $head_1$, 3) * list(L_2, $head_2$, m)}
⟺ {list(a_1; a_2; a_3, $head_1$, 3) * list(L_2, $head_2$, m)}

Also, using the list predicate, we can re-write the pre-condition as:
{list(L, *head*, m+3)}
⟺ {list(L_1; L_2, *head*, 3+m)}
⟺ {list(a_1; a_2; a_3; L_2, *head*, 3+m)}

This just says that list L is comprised of two sub-lists L_1 and L_2 where we might want the first 3 elements (whatever their values be) denoted by a_1, a_2, and a_3 to be as part of (sub)list L_1.

Full proof outline:

{list(L, *head*, m+3)}
⟺{list(L_1; L_2, *head*, 3+m)}
⟺{list(a_1; a_2; a_3; L_2, *head*, 3+m)}

$$\Longrightarrow \{head \mapsto a_1, i_1 * i_1 \mapsto a_2, i_2 * i_2 \mapsto a_3, i_3 * list(L_2, i_3, m)\}$$

$head_1 := head;$

$$\{head_1 \mapsto a_1, i_1 * i_1 \mapsto a_2, i_2 * i_2 \mapsto a_3, i_3 * list(L_2, i_3, m)\}$$

$x_0 := !\,(head_1 + 1);$

$$\{x_0 = i_1 \wedge head_1 \mapsto a_1, x_0 * i_1 \mapsto a_2, i_2 * i_2 \mapsto a_3, i_3 * list(L_2, i_3, m)\}$$
$$\Longrightarrow \{head_1 \mapsto a_1, x_0 * x_0 \mapsto a_2, i_2 * i_2 \mapsto a_3, i_3 * list(L_2, i_3, m)\}$$

$x_1 := !\,(x_0 + 1);$

$$\{x_1 = i_2 \wedge head_1 \mapsto a_1, x_0 * x_0 \mapsto a_2, x_1 * i_2 \mapsto a_3, i_3 * list(L_2, i_3, m)\}$$
$$\Longrightarrow \{head_1 \mapsto a_1, x_0 * x_0 \mapsto a_2, x_1 * x_1 \mapsto a_3, i_3 * list(L_2, i_3, m)\}$$

$head_2 := !\,(x_1 + 1);$

$$\{head_2 = i_3 \wedge head_1 \mapsto a_1, x_0 * x_0 \mapsto a_2, x_1 * x_1 \mapsto a_3, i_3 * list(L_2, i_3, m)\}$$
$$\Longrightarrow \{head_1 \mapsto a_1, x_0 * x_0 \mapsto a_2, x_1 * x_1 \mapsto a_3, head_2 * list(L_2, head_2, m)\}$$

$!\,(x_1 + 1) \coloneqq nil$

$$\{head_1 \mapsto a_1, x_0 * x_0 \mapsto a_2, x_1 * x_1 \mapsto a_3, nil * list(L_2, head_2, m)\}$$

$$\Longrightarrow \{list(a_1; a_2; a_3, head_1, 3) * list(L_2, head_2, m)\}$$

⟺{list(L_1, $head_1$, 3) * list(L_2, $head_2$, m)}

As per the HW pdf, there is no need to justify the proof obligations. I can provide the proof for these if needed.

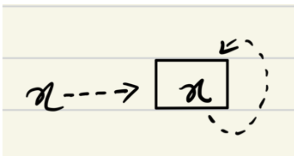
2. Resource Logic

Task 2.1

a) $x \mapsto x$

Ans.

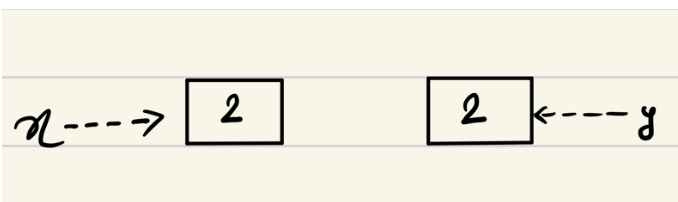
It's a precise assertion.



b) $x \mapsto 2 * y \mapsto 2$

Ans.

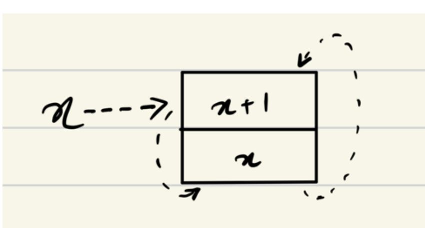
It's a precise assertion and two memory locations which are separable.



c) $x \mapsto x + 1,\ x$

Ans.

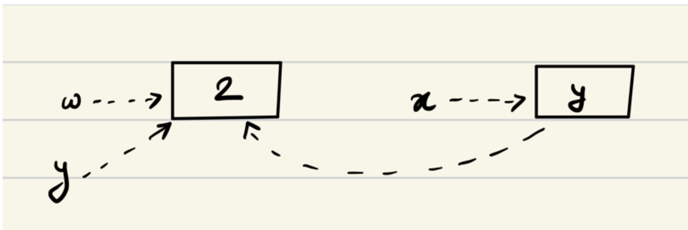
This is a precise assertion with exactly two adjacent heap elements.



d) $w \mapsto 2 * (x \mapsto y \wedge y = w)$

Ans.

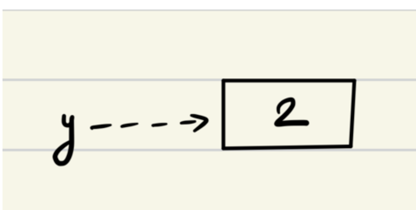
It's a precise assertion. Also, the heap must satisfy $x \mapsto y$ and $y = w$.



e) $(\exists x. y \mapsto x) \wedge y \mapsto 2$

Ans.

This is a precise assertion which must satisfy $\exists x. y \mapsto x$, which suggests that one such value of x can be 2.

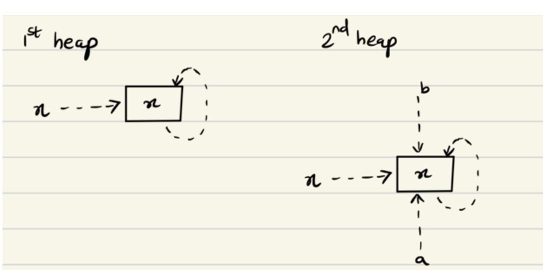


Task 2.2

a) $x \hookrightarrow x$

Ans.

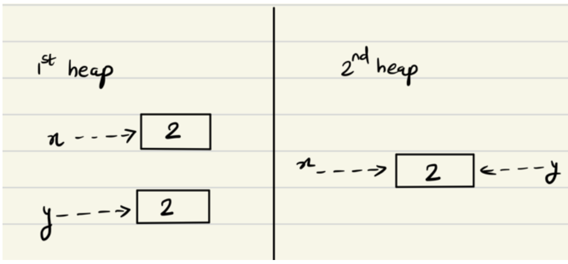
It's an imprecise assertion and we need to draw two heaps which satisfy the above resource logic.



b) $x \hookrightarrow 2 \wedge y \hookrightarrow 2$

Ans.

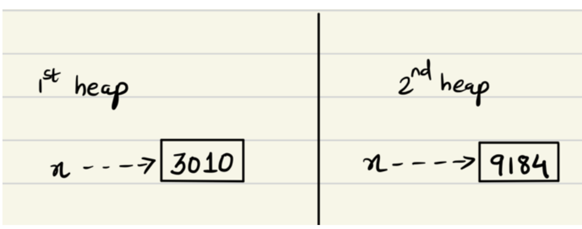
It's an imprecise assertion.



c) $x \mapsto -$

Ans.

As per the notes, we write $e \mapsto -$ when the heap has exactly one location e but its value is not important for us. It is defined as $\exists x.\, e \mapsto x$, where x is not free in e.



Task 2.3

a) $(x \mapsto 2 * y \mapsto 2) * z \mapsto 2 \Longrightarrow (z \mapsto 2 * y \mapsto 2) * x \mapsto 2$

Ans.

$$(x \mapsto 2 * y \mapsto 2) * z \mapsto 2$$
$$\Leftrightarrow x \mapsto 2 * (y \mapsto 2 * z \mapsto 2) \quad \text{...Using Rule 2: } (p_1 * p_2) * p_3 \Leftrightarrow p_1 * (p_2 * p_3)$$
$$\Leftrightarrow (y \mapsto 2 * z \mapsto 2) * x \mapsto 2 \quad \text{...Using Rule 1: } (p_1 * p_2) \Leftrightarrow (p_2 * p_1)$$
$$\Leftrightarrow (z \mapsto 2 * y \mapsto 2) * x \mapsto 2 \quad \text{...Using Rule 1: } (p_1 * p_2) \Leftrightarrow (p_2 * p_1)$$

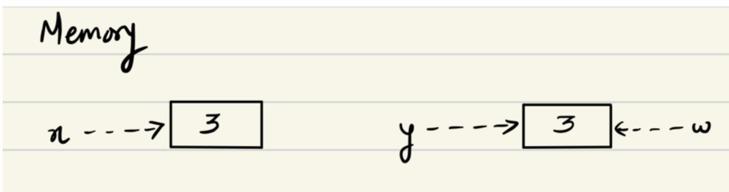
Hence, $(x \mapsto 2 * y \mapsto 2) * z \mapsto 2 \Longrightarrow (z \mapsto 2 * y \mapsto 2) * x \mapsto 2$

b) $x \hookrightarrow 3 \wedge y \hookrightarrow 3 \Longrightarrow x = y$

Ans.

This is an example of imprecise assertion. According to imprecise assertion definition in the notes $e \hookrightarrow e'$, it indicates that somewhere in the heap, e points to e'.
In our question, x=y asserts that both are same pointers and point to same location in the heap which is not true as per the imprecise assertion.

Hence, providing a counterexample:



Here, we can see that even if somewhere in the memory, $x \hookrightarrow 3 \wedge y \hookrightarrow 3$, it does not imply $x = y$

Hence, $x \hookrightarrow 3 \wedge y \hookrightarrow 3 \not\Leftarrow x = y$

c) **emp** * $\big((\exists x. y \mapsto x) * w \mapsto 2\big) \Longrightarrow \exists x. (y \mapsto x * w \mapsto 2)$

Ans.

emp * $\big((\exists x. y \mapsto x) * w \mapsto 2\big)$
$\Leftrightarrow \big((\exists x. y \mapsto x) * w \mapsto 2\big)$ * **emp** …Using Rule 1: $(p_1 * p_2) \Leftrightarrow (p_2 * p_1)$
$\Leftrightarrow \big((\exists x. y \mapsto x) * w \mapsto 2\big)$ …Using Rule 3: $(p * emp) \Leftrightarrow (p)$
$\Leftrightarrow \exists x. (y \mapsto x * w \mapsto 2)$ …Using Rule 6: $(\exists x. p_1) * p_2 \Leftrightarrow (\exists x. p_1 * p_2)$
as x is not free in $w \mapsto 2$, we can use Rule 6

Hence, **emp** * $\big((\exists x. y \mapsto x) * w \mapsto 2\big) \Longrightarrow \exists x. (y \mapsto x * w \mapsto 2)$

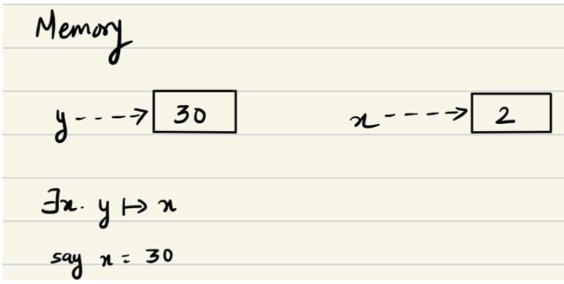
d) $(\exists x. y \mapsto x) * x \mapsto 2 \Longrightarrow \exists x. (y \mapsto x * x \mapsto 2)$

Ans.

$(\exists x. y \mapsto x) * x \mapsto 2$
To apply rule 6: $(\exists x. p_1) * p_2 \Leftrightarrow (\exists x. p_1 * p_2)$, x should not be free in p_2.

In our problem, the x outside is not bounded by the existential quantifier. This means x is free. Hence, we cannot apply Rule 6 here.

Hence, we must provide a counterexample:



Here, it can be clearly seen a case where $(\exists x. y \mapsto x) * x \mapsto 2 \nRightarrow \exists x. (y \mapsto x * x \mapsto 2)$

3. Nondeterminism

Task 3.1

a) Given: $S_1 \triangleq while \left\{ x \geq 0 \rightarrow y \coloneqq \frac{x}{y} \quad x \leq 0 \rightarrow y \coloneqq y * x \right\}$

$$\sigma \triangleq \{x = 0, y = 1\}$$

Ans.
One version of non-deterministic small step semantics of this problem can be:
$< S_1, \{x = 0, y = 1\} >$
$\rightarrow < y \coloneqq y * x;\ S_1, \{x = 0, y = 1\} >$
$\rightarrow^2 < S_1, \{x = 0, y = 0\} >$
$\rightarrow < y \coloneqq y * x;\ S_1, \{x = 0, y = 0\} >$
$\rightarrow^2 < S_1, \{x = 0, y = 0\} >$
$\rightarrow < y \coloneqq y * x;\ S_1, \{x = 0, y = 0\} >$
$\rightarrow^2 < S_1, \{x = 0, y = 0\} >$
…

The execution continues like this.

Another version of non-deterministic small step semantics of this problem can be:
$< S_1, \{x = 0, y = 1\} >$
$\rightarrow < y \coloneqq \frac{x}{y};\ S_1, \{x = 0, y = 1\} >$
$\rightarrow^2 < S_1, \{x = 0, y = 0\} >$
$\rightarrow < y \coloneqq y * x;\ S_1, \{x = 0, y = 0\} >$
$\rightarrow^2 < S_1, \{x = 0, y = 0\} >$
$\rightarrow < y \coloneqq y * x;\ S_1, \{x = 0, y = 0\} >$
$\rightarrow^2 < S_1, \{x = 0, y = 0\} >$
…

The execution continues like this.

b) Given: $S_1 \triangleq while \left\{x \geq 0 \rightarrow y \coloneqq \frac{x}{y} \quad x \leq 0 \rightarrow y \coloneqq y * x\right\}$

$$\sigma \triangleq \{x = 0, y = 1\}$$

Ans: Big-step semantics for nondeterministic loops involves the same technique used for regular while loops.

$M(S_1, \sigma) = \sum$ (set of states)
$M(while\ e\ do\ S\ od, \sigma) = \sum_k$

$\sum_0 = \{\{x = 0, y = 1\}\}$

$\sum_1 = M\left(y \coloneqq \frac{y}{x}, \sigma\right) \vee M(y \coloneqq y * x, \sigma)$

$= M\left(y \coloneqq \frac{y}{x}, \{x = 0, y = 1\}\right) \vee M(y \coloneqq y * x, \{x = 0, y = 1\})$
$= \left\{\{x = 0, y = 0\}\right\} \vee \{\{x = 0, y = 0\}\}$
$= \left\{\{x = 0, y = 0\}\right\}$

$\sum_2 = M\left(y \coloneqq \frac{y}{x}, \sigma\right) \vee M(y \coloneqq y * x, \sigma)$

$= M\left(y \coloneqq \frac{y}{x}, \{x = 0, y = 0\}\right) \vee M(y \coloneqq y * x, \{x = 0, y = 0\})$
$= \{\perp\} \vee \{\{x = 0, y = 0\}\}$
$= \left\{\perp, \{x = 0, y = 0\}\right\}$

4. Parallel Programs

Task 4.1

Given: $S_2 \triangleq [x \coloneqq y; y \coloneqq y + \overline{1} \parallel y \coloneqq y + \overline{3}]$

$\sigma \triangleq \{x = 3, y = 2\}$ and heap h

Ans:

Let's say that the given heap $h = \{\}$

$$< [x \coloneqq y; y \coloneqq y + \bar{1} \parallel y \coloneqq y + \bar{3}], (\{x = 3, y = 2\}, \{\}) >$$

$$< [y \coloneqq y + \bar{1} \parallel y \coloneqq y + \bar{3}], (\{x = 2, y = 2\}, \{\}) >$$

$$< [x \coloneqq y; y \coloneqq y + \bar{1} \parallel skip], (\{x = 3, y = 5\}, \{\}) >$$

$$< [skip \parallel y \coloneqq y + \bar{3}], (\{x = 2, y = 3\}, \{\}) >$$

$$< [y \coloneqq y + \bar{1} \parallel skip], (\{x = 2, y = 5\}, \{\}) >$$

$$< [y \coloneqq y + \bar{1} \parallel skip], (\{x = 5, y = 5\}, \{\}) >$$

$$< [skip \parallel skip], (\{x = 2, y = 6\}, \{\}) >$$

$$< [skip \parallel skip], (\{x = 5, y = 6\}, \{\}) >$$

$$M(S_2, (\{x = 3, y = 2\}, \{\}) = \{(\{x = 2, y = 6\}, \{\}), (\{x = 5, y = 6\}, \{\})\}$$

This is the evaluation graph and the final output.

5. One more wrap-up question.

Task 5.1

Ans. Totally, I spent 18 hours on this assignment. 16 hours on solving and 2 hours to type this.