6 Hoare Logic and Model Checking (cp526)

Consider a programming language with commands $C$ consisting of the skip no-op command, sequential composition $C_1; C_2$, loops while $B$ do $C$ for Boolean expressions $B$, conditionals if $B$ then $C_1$ else $C_2$, assignment $X := E$ for program variables $X$ and arithmetic expressions $E$, heap allocation $X := \text{alloc}(E_1, \ldots, E_n)$, heap assignment $[E_1] := E_2$, heap dereference $X := [E]$, and heap location disposal $\text{dispose}(E)$. Assume null = 0, and predicates for lists and partial lists:

$$\text{list}(t, []) = (t = \text{null}) \land \text{emp}$$
$$\text{list}(t, h : : \alpha) = \exists y. (t \mapsto h) * ((t + 1) \mapsto y) * \text{list}(y, \alpha)$$
$$\text{plist}(t_1, [], t_2) = (t_1 = t_2) \land \text{emp}$$
$$\text{plist}(t_1, h : : \alpha, t_2) = \exists y. (t_1 \mapsto h) * ((t_1 + 1) \mapsto y) * \text{plist}(y, \alpha, t_2)$$

In the following, all triples are linear separation logic triples. No proofs are required.

(a) Precisely describe a stack and a heap that satisfy $X \mapsto Y * Y \mapsto X$. Give a (non-looping) command $C$ that satisfies the following triple.

$$\{\text{emp}\} C \{X \mapsto Y * Y \mapsto X\}.$$ [3 marks]

(b) Define and explain a partial correctness rule for a new command $\text{unseq}(C_1; C_2)$, which executes commands $C_1$ and $C_2$ in either order ($C_1; C_2$ or $C_2; C_1$). Maintain soundness of the proof system, and ensure the rule accurately reflects the behaviour of the new command. [3 marks]

(c) Do the same for a new command $\text{add_to}(E_1, E_2)$. If expressions $E_1$ and $E_2$ evaluate to allocated, disjoint memory locations, it increments the value stored at the first location by the value stored at the second. Otherwise it crashes. [3 marks]

For each of the following triples, give a loop invariant that would prove it.

(d) This command duplicates each list element. As per precondition assume $Y$ is initially the head $X$; assume dup duplicates elements, e.g. dup [1, 2] = [1, 1, 2, 2].

$$\{\text{list}(X, \alpha) \land Y = X\}
\text{while } Y \neq \text{null} \text{ do } (V := [Y]; N := [Y+1]; D := \text{alloc}(V, N); [Y+1] := D; Y := N)\{\text{list}(X, \text{dup } \alpha)\}$$ [4 marks]

(e) This command removes all negative numbers in a list, assuming it starts with 0.

$$\{\text{list}(X, [0]++\alpha)\}
L := X; Y := [X+1];
\text{while } Y \neq \text{null} \text{ do } (V := [Y]; N := [Y+1];
\text{if } V < 0 \text{ (dispose}(Y); \text{dispose}(Y+1)) \text{ else } ([L+1] := Y; L := Y) ; Y := N)
; L+1 := \text{null} \{\text{list}(X, [0]++(\text{remove_negatives } \alpha))\}$$ [7 marks]