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, assigment X `:=` E for program variables X and arithmetic expressions E, heap allocation X `:= alloc(`E_1`,...,`E_n`)`, heap assignment `[`E_1`] :=` E_2, heap dereference X `:= [`E`]`, and heap location disposal `dispose(`E`)`. Assume `null` $= 0$, and predicates for lists and partial lists:

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

In the following, all triples are linear separation logic triples.

(*a*) Explain why a command C of your choice satisfies the following triple, or explain why no such C exists: $\{\texttt{null} \mapsto 5\}\ C\ \{\top\}$. [2 marks]

(*b*) Explain why a command C of your choice satisfies the following triple (i.e. moves v to a different location): $\{x \mapsto v \land X = x\}\ C\ \{Y \mapsto v \land Y \neq x\}$. [2 marks]

(*c*) Give a loop invariant that would serve to prove the following triple, for a command that creates a reversed copy of a list (no proof outline required).
$\{\text{list}(X, \alpha)\}$
`Y := null; C := X;`
`while C` $\neq$ `null do (V := [C]; Y := alloc(V,Y); C := [C+1])`
$\{\text{list}(X, \alpha) * \text{list}(Y, \text{rev}\ \ \alpha)\}$ [4 marks]

(*d*) Adjust the program in (*c*) with a new loop body C_L, so it (still) terminates and $\{\text{list}(X, \alpha)\}$ `Y := null; C := X; while C` $\neq$ `null do` C_L $\{\text{list}(Y, \text{rev}\ \ \alpha)\}$ holds (no proof, loop invariant, or termination argument required). [2 marks]

(*e*) Consider an *unsound* extension of the separation-logic proof system with the rule $\{E_1 > 0 \land \text{emp}\}$ `alloc_here`(E_1, E_2) $\{E_1 \mapsto E_2\}$ for a new command `alloc_here`(E_1, E_2). Explain in detail, with reference to the proof rules, how $\{\text{emp}\}\ C\ \{\bot\}$ is derivable, for a non-looping C of your choice. [4 marks]

(*f*) Give a loop invariant that would serve to prove the following triple, for a command that creates a list of the Fibonacci numbers up to n (no proof outline required). Assume $\text{fibs}(i, j) = [\text{fib}\ i, \ldots, \text{fib}\ j]$ for $i \leq j$ and $[]$ otherwise.
$\{\text{emp} \land (N = n \land n > 2)\}$
`II := alloc(1,null); I := alloc(0,II); X := I; C := 2;`
`while C` $\leq$ `N do (IV := [I]; IIV := [II]; I := II;`
`II := alloc(IV+IIV,null); [I+1] := II; C := C+1)`
$\{\text{list}(X, \text{fibs}(0, n))\}$ [6 marks]