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 `:= 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:

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

In the following, all triples are linear separation logic triples. 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.
$\{emp\}\ C\ \{X \mapsto Y * Y \mapsto X\}$. [3 marks]

(*b*) Define and explain a partial correctness rule for a new command `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 `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 $[1,2] = [1,1,2,2]$.
$\{\text{list}(X, \alpha) \wedge Y = X\}$

```
while Y≠null do (V:=[Y]; N:=[Y+1]; D:=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.
$\{\text{list}(X, [0]\text{++}\alpha)\}$

```
L:=X; Y:=[X+1];
while Y≠null do (
  V:=[Y]; N:=[Y+1];
  (if V<0 (dispose(Y); dispose(Y+1)) else ([L+1]:=Y; L:=Y)); Y:=N
); [L+1]:=null
```

$\{\text{list}(X, [0]\text{++}(\text{remove\_negatives}\ \ \alpha))\}$ [7 marks]