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:

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

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

(*a*) Find a command C satisfying the following separation logic partial correctness triple: $\{\top\}\ C\ \{X \mapsto 0 * X \mapsto 0\}$. [2 marks]

(*b*) Give a loop invariant that would serve to prove the following triple, where 'map negate α' is the list of negated values in α (no proof outline required):
$\{\mathrm{list}(X, \alpha)\}$
`Y = X; while Y` $\neq$ `null do (V := [Y]; [Y] = V * (-1); Y = [Y + 1])`
$\{\mathrm{list}(X, \mathrm{map\ \ negate\ \ }\alpha)\}$ [4 marks]

(*c*) Give a loop invariant that would serve to prove the following triple, for a program that finds the last element of a list (no proof outline required):
$\{\mathrm{list}(X, \alpha \texttt{ ++ } [l])\}$

```
CUR = X; NEXT = [X + 1];
while NEXT ≠ null do (CUR = NEXT; NEXT = [NEXT + 1]);
LAST = [CUR]
```

$\{\mathrm{list}(X, \alpha \texttt{ ++ } [l]) \wedge LAST = l\}$ [5 marks]

(*d*) Explain why a proof of Part (*c*) would not succeed if the post-condition of the triple was replaced with $\{emp \wedge LAST = l\}$. [3 marks]

(*e*) Give a loop invariant that would serve to prove the following triple, for a program that copies a given list (no proof outline required):
$\{\mathrm{list}(X, \alpha) \wedge \alpha \neq []\}$

```
V = [X]; Y := alloc(V, null); CUR := [X+1]; OLD = Y;
while CUR ≠ null do (
   V = [CUR]; N = alloc(V, null); [OLD + 1] = N;
   CUR = [CUR + 1]; OLD = N
)
```

$\{\mathrm{list}(X, \alpha) * \mathrm{list}(Y, \alpha)\}$ [6 marks]