7 Hoare Logic and Model Checking (NA)

Consider a programming language that consists of commands `C` composed from assignments `V := E` (where `V` is a program variable and `E` is an expression), the no-op `skip`, sequencing `C1;C2`, conditionals `if B then C1 else C2` (where `B` is a boolean expression), and loops `while B do C`.

(*a*) Explain informally what it means for a partial correctness triple $\{P\}$ `C` $\{Q\}$ to be valid. [2 marks]

(*b*) Consider the partial correctness triple $\{\top\}$ `C` $\{\bot\}$ (where $\top$ is the true assertion, and $\bot$ is the false assertion). Give a command `C` that makes the triple valid or explain why no such command exists. [2 marks]

(*c*) Consider a new primitive command `either C1 C2` which non deterministically executes either one of its arguments: `C1` or else `C2`. Give a partial correctness logic rule for such a command, maintaining soundness and relative completeness. Give an alternative partial correctness logic rule for such a command, maintaining soundness but *not* relative completeness. [2 marks]

(*d*) Consider a new command `flip V` which randomly assigns either 0 or 1 to the variable `V`. Give a logic rule for partial correctness for such a command, maintaining soundness and relative completeness. Define `flip` using `either` from Part (*c*). [2 marks]

(*e*) Consider a new primitive command `havoc V` which assigns a random integer to the variable `V`. Give a logic rule for partial correctness for such a command, maintaining soundness and relative completeness. [2 marks]

(*f*) Consider the program `Z:=0; while (Z` $\neq$ `X` $\wedge$ `Z` $\neq$ `Y) do Z := Z+1`. Give a reasonable pre-condition so that the program terminates with `Z` equal to the minimum of `X` and `Y`. Propose an invariant for the while loop, and use it to prove that the program satisfies its partial correctness specification. [5 marks]

(*g*) Consider an extension of our programming language above with heap assignment `[E1] := E2`, heap dereference `X := [E2]`, and disposal of heap locations `dispose(E)`. Recall the list representation predicate

$$list(t, []) = (t = \texttt{null})$$
$$list(t, h :: \alpha) = (\exists y.\, t \mapsto h * (t+1) \mapsto y * list(y, \alpha))$$

Consider the following program that deallocates a list, and counts how many list elements it deallocated:

```
while (X≠null) do (N:=N+1; Y:=[X+1]; dispose(X); dispose(X+1); X:=Y)
```

Propose an invariant for the loop that, given precondition $N = 0 \wedge list(\texttt{X}, \alpha)$, is sufficient to establish the postcondition $\texttt{N} = length(\alpha) \wedge list(\texttt{X}, [])$. [5 marks]