Exercise 1F-2

I agree with Hoare's assertion that a good programming language should encourage the programmer to write code that is self-documenting. I find strong type systems are excellent tools that "force" programmers to provide information about their functions that add documentation and improve readability. E.g. I believe ML-like type systems improve readability. For example, in ML, a function f may be annotated as $f : \texttt{List}\ \alpha \rightarrow \texttt{Int}$. This simple annotation, which the programmer is forced to provide, tells us this function is stateless and takes a list with elements of type α, and returns an integer. Without looking at the function definition, we can reasonably guess the meaning of f. Some kind of length function that counts the elements of the input.

However, I think type systems sometimes can make programmers less readable. Like mathematical notation, programmers can introduce new terms and types that readers unfamiliar with these terms will only be confused by. An infamous example is the Lens library in the Haskell programming language. A user must understand each of the many types used in Lens functions to be able to use library in the first place. Perhaps that is by construction, but we never need such in an in-depth knowledge to use libraries from Python or C++.

Exercise 1F-3

We are only adding the division operator, so none of the existing operational semantics in Aexp need to be changed. Since only integers are defined in IMP, we will only add an *integer* division operator. Integer division is simply division but discarding the remainder. There is an additional caution of the divide-by-zero case. In our operational semantic, we will simply leave dividing by zero undefined.

We give the following operational semantic:

$$\frac{\langle e_1, \sigma \rangle \Downarrow n_1 \quad \langle e_2, \sigma \rangle \Downarrow n_2 \quad n_2 \neq 0}{\langle e_1 / e_2, \sigma \rangle \Downarrow n_1 // n_2}$$

The symbol // represents the integer division operator on integers.

Exercise 1F-4

$$\frac{\langle e, \sigma_1 \rangle \Downarrow n \quad \langle c, \sigma_1[x := n] \rangle \Downarrow \sigma_2}{\langle \texttt{let}\ \ x = e\ \ \texttt{in}\ \ c, \sigma_1 \rangle \Downarrow \sigma_2[x := \sigma_1(x)]}$$

Exercise 1F-4

We give the extensions to redux and context rules for IMP:

$$H ::= \ldots \mid \texttt{let}\ \ x = H\ \ \texttt{in}\ \ c \tag{1}$$

$$r ::= \ldots \mid \texttt{let}\ \ x = n\ \ \texttt{in}\ \ c \tag{2}$$

and the following reduction rule:

$$\langle \texttt{let}\ \ x = n\ \ \texttt{in}\ \ c, \sigma \rangle \rightarrow \langle x := n; c; x := \sigma(x), \sigma \rangle \tag{3}$$

1 HW1 (select all pages: your first page has your name and bookkeeping, and all others are anonymous))

- 0 pts Correct

Peer Review ID: 62718541 — enter this when you fill out your peer evaluation via gradescope