2. Language Design

Paragraph 1: After reading, I come up with the time that I learned QBasic in 2005 when I was in Elementary school. That is my first time to learn programming. I found it really tough to handon → terrible UI design, hard to read, and poor syntax. But later in high school, I find my interest in programming in Java, since its nice UI design and the readable code structure.

Program 2: One point favor of: Readability

In the reading, the author said it would be wise for the designer of programming language to concentrate on the easier task of designing a readable language to begin with, this is because the evidence shows there it is unlikely the output from computer is more readable than its input. I think it is very reasonable, since based on my programming experience, a readable programming language can make a huge change: in QBasic I almost gave up since the readability; in Java, I make the decision to make CSE as my major since the easy, and readable language.

Program 3: One point against of: Language Feature Design

In the reading, it said the designer of a new feature should concentrate on one feature at a time. Author also said that the designer should make sure that his feature should mitigates some disadvantages or remedies some incompleteness, of the language without compromising any of its existing merits. I disagree with this, since there are lots of features that may not designed for solving the disadvantages or incompleteness. Some features is for universalization, and others may contribute to the diversity of one operation. For example, we have list_a + list_b, we also have list_a.append(i for i in list_b), before append() is created, the language is complete and no disadvantages, and the append() only improves the diversity of one operation/method, it lets users more options to select.

3. Simple Operational Semantics

$$\frac{\langle e_1, \sigma \rangle \Downarrow n_1 \quad \langle e_2, \sigma \rangle \Downarrow n_2}{\langle e_1 / e_2, \sigma \rangle \Downarrow \text{Error}} \quad \text{when } n_2 == 0$$

$$\frac{\langle e_1, \sigma \rangle \Downarrow n_1 \quad \langle e_2, \sigma \rangle \Downarrow n_2}{\langle e_1 / e_2, \sigma \rangle \Downarrow n_1 / n_2} \quad \text{when } n_2 \text{ divides } n_1$$

$$\frac{\langle e_1, \sigma \rangle \Downarrow n_1 \quad \langle e_2, \sigma \rangle \Downarrow n_2}{\langle e_1 / e_2, \sigma \rangle \Downarrow (n_1 - n_1 \bmod n_2)/n_2} \quad \text{when otherwise}$$

* Note that in Ocaml, $-3 \bmod 2 = -1$.
so we want $5/2 = 1$ and $-5/2 = -2$,
checks $(5 - 5 \bmod 2)/2 = (5-1)/2 = 2$
$(-5 - (-5) \bmod 2)/2 = (-5-(-1))/2 = -2$ works.

4. Language Feature Design, Large Step

$$\frac{\langle e, \sigma \rangle \Downarrow n \quad \langle c, \sigma[x := n] \rangle \Downarrow \sigma'}{\langle \text{let } x = e \text{ in } c, \sigma \rangle \Downarrow \sigma'[x := \sigma(x)]}$$

5. Small-step

let $X := n$ in C with initial state $[X := ori]$

$\Downarrow$

$\langle tmp := X ; X := n ; C ; X := tmp \rangle$

$\langle$ Comm, State$\rangle$	Redex •	Context
$\langle tmp := X ; X := n ; C ; X := tmp, [X := ori] \rangle$	X	$tmp := \bullet ; X := n ; C ; X := tmp$
$\langle tmp := ori ; X := n ; C ; X := tmp, [X := ori] \rangle$	$tmp := ori$	$\bullet ; X := n ; C ; X := tmp$
$\langle skip ; X := n ; C ; X := tmp, \begin{bmatrix} X := ori \\ tmp := ori \end{bmatrix} \rangle$	$skip ; X := n ; C ; X := tmp$	$\bullet$
$\langle X := n ; C ; X := tmp, \begin{bmatrix} X := ori \\ tmp := ori \end{bmatrix} \rangle$	$X := n$	$\bullet ; C ; X := tmp$
$\langle skip ; C ; X := tmp, \begin{bmatrix} X := n \\ tmp := ori \end{bmatrix} \rangle$	$skip ; C ; X := tmp$	$\bullet$
$\langle C ; X := tmp, \begin{bmatrix} X := n \\ tmp := ori \end{bmatrix} \rangle$	C	$\bullet ; X := tmp$
$\langle skip ; X := tmp, \begin{bmatrix} X := n \\ tmp := ori \end{bmatrix} \rangle$	$skip ; X := tmp$	$\bullet$
$\langle X := tmp, \begin{bmatrix} X := n \\ tmp := ori \end{bmatrix} \rangle$	tmp	$X := \bullet$
$\langle X := ori, \begin{bmatrix} X := n \\ tmp := ori \end{bmatrix} \rangle$	$X := ori$	$\bullet$
$\langle skip, \begin{bmatrix} X := ori \\ tmp := ori \end{bmatrix} \rangle$		

So: Generally: $l\text{-}l ::= \ldots \mid let\ x = l\text{-}l\ in\ c$

$r ::= \ldots \mid let\ X = n\ in\ C$

$\langle let\ X := n\ in\ C, \sigma \rangle \rightarrow \langle tmp := X ; X := n ; C ; X := tmp, \sigma' \rangle$

Then As shown in the table

$\langle let\ X := n\ in\ C, \sigma \rangle \rightarrow \langle skip ; C ; X := \sigma(X), \sigma[X := n] \rangle \rightarrow \cdots \rightarrow \langle skip ; \sigma[X := \sigma(X)] \rangle$

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

- **0 pts** Correct