CS320 Programming Languages
Exercise #6: SRBFAE

The goal of Exercise #6 is to implement the interpreter of `SRBFAE`, which is the extended version of `BFAE` with general Sequences and Records. The concrete syntax of `SRBFAE` is written in the extended Backus-Naur form (EBNF). Note that `{ }` denotes a repetition of zero or more times.

1 Improving Sequences

Generalize sequences to allow one or more sub-expressions, instead of exactly two sub-expressions:

```
expr  ::=  num
         |  "(" expr "+" expr ")"
         |  "(" expr "-" expr ")"
         |  id
         |  "{" id "=>" expr "}"
         |  expr "(" expr ")"
         |  "Box" "(" expr ")"
         |  expr "." "set" "(" expr ")"
         |  expr "." "get"
         |  "{" expr {";" expr} "}"
```

For example:

```
test(run("""{
              b => {
                b.set((2 + b.get));
                b.set((3 + b.get));
                b.set((4 + b.get));
                b.get
              }
            }(Box(1))"""), "10")
```

2 Records

Extend your interpreter to support the construction of records with named fields, and to support field selection and field update from a record:

```
expr  ::=  ...
         |  "{" "}"
         |  "{" id "=" expr {"," id "=" expr} "}"
         |  expr "." id
         |  "{" expr "." id "=" expr "}"
```

`{` e_1`.`x `=` e_2 `}` changes within the record produced by e_1 the value of the field named by x; the value of e_2 determines the field's new value, and that value is also the result of the expression. Note that changes within the

record using the field name that does not exist in the record print error messages containing `"no such field"`. A record is a mutable data structure as same as a box. For example:

```
testExc(run("{ x = 1 }.y"), "no such field")
test(run("""{
              r => {
                { r.x = 5 };
                r.x
              }
            }({ x = 1 })"""), "5")
```

The examples below use `run`, which returns a number string if the interpreter produces a number, returns the string `"function"` if the interpreter produces a function value, returns the string `"box"` if the interpreter produces a box value, and returns the string `"record"` if the interpreter produces a record value:

```
test(run("42"), "42")
test(run("{ x => x }"), "function")
test(run("Box(1)"), "box")
test(run("{}"), "record")
```

3 Abstract Syntax

For abstract syntax, we use $\cdots$ to denote repetitions.

$$
\begin{array}{rcl}
e & ::= & n \\
& | & e + e \\
& | & e - e \\
& | & x \\
& | & \lambda x.e \\
& | & e\ e \\
& | & \mathsf{ref}\ e \\
& | & e{:=}e \\
& | & !e \\
& | & e;\cdots;e \\
& | & \{x = e, \cdots, x = e\} \\
& | & e.x \\
& | & e.x{:=}e
\end{array}
\qquad
\begin{array}{rcl}
v & ::= & n \\
& | & \langle \lambda x.e, \sigma \rangle \\
& | & a \\
& | & \{x = a, \cdots, x = a\}
\end{array}
\qquad
\begin{array}{rcl}
n & \in & \mathbb{Z} \\
x & \in & \mathit{Var} \\
a & \in & \mathit{Addr} \\
v & \in & \mathit{Val} \\
\sigma & \in & \mathit{Var} \to \mathit{Val}
\end{array}
$$

4 Operational Semantics

$\boxed{\sigma, M \vdash e \Rightarrow v, M}$

$$\sigma, M \vdash n \Rightarrow n, M$$

$$\frac{\sigma, M \vdash e_1 \Rightarrow n_1, M_1 \quad \sigma, M_1 \vdash e_2 \Rightarrow n_2, M_2}{\sigma, M \vdash e_1 + e_2 \Rightarrow n_1 + n_2, M_2}$$

$$\frac{\sigma, M \vdash e_1 \Rightarrow n_1, M_1 \quad \sigma, M_1 \vdash e_2 \Rightarrow n_2, M_2}{\sigma, M \vdash e_1 - e_2 \Rightarrow n_1 - n_2, M_2}$$

$$\frac{x \in \mathit{Domain}(\sigma)}{\sigma, M \vdash x \Rightarrow \sigma(x), M}$$

$$\sigma, M \vdash \lambda x.e \Rightarrow \langle \lambda x.e, \sigma \rangle, M$$

$$\frac{\sigma, M \vdash e_1 \Rightarrow \langle \lambda x.e, \sigma' \rangle, M_1 \quad \sigma, M_1 \vdash e_2 \Rightarrow v', M_2 \quad \sigma'[x \mapsto v'], M_2 \vdash e \Rightarrow v, M_3}{\sigma, M \vdash e_1\ e_2 \Rightarrow v, M_3}$$

$$\frac{\sigma, M \vdash e \Rightarrow v, M_1 \quad a \notin \mathit{Domain}(M_1)}{\sigma, M \vdash \mathsf{ref}\ e \Rightarrow a, M_1[a \mapsto v]}$$

$$\frac{\sigma, M \vdash e_1 \Rightarrow a, M_1 \quad \sigma, M_1 \vdash e_2 \Rightarrow v, M_2}{\sigma, M \vdash e_1 := e_2 \Rightarrow v, M_2[a \mapsto v]}$$

$$\frac{\sigma, M \vdash e \Rightarrow a, M_1 \quad a \in \mathit{Domain}(M_1)}{\sigma, M \vdash !e \Rightarrow M_1(a), M_1}$$

$$\frac{\sigma, M_0 \vdash e_1 \Rightarrow v_1, M_1 \quad \cdots \quad \sigma, M_{n-1} \vdash e_n \Rightarrow v_n, M_n}{\sigma, M_0 \vdash e_1; \cdots; e_n \Rightarrow v_n, M_n}$$

$$\frac{\begin{array}{c}\sigma, M_0 \vdash x_1 \Rightarrow v_1, M_1' \quad \cdots \quad \sigma, M_{n-1} \vdash x_n \Rightarrow v_n, M_n' \\ a_1 \notin \mathit{Domain}(M_1') \quad \cdots \quad a_n \notin \mathit{Domain}(M_n') \qquad M_1 = M_1'[a_1 \mapsto v_1] \quad \cdots \quad M_n = M_n'[a_n \mapsto v_n]\end{array}}{\sigma, M_0 \vdash \{x_1 = e_1, \cdots, x_n = e_n\} \Rightarrow \{x_1 = a_1, \cdots, x_n = a_n\}, M_n}$$

$$\frac{\sigma, M \vdash e \Rightarrow \{\cdots, x = a, \cdots\}, M_1 \quad a \in \mathit{Domain}(M_1)}{\sigma, M \vdash e.x \Rightarrow M_1(a), M_1}$$

$$\frac{\sigma, M \vdash e_1 \Rightarrow \{\cdots, x = a, \cdots\}, M_1 \quad \sigma, M_1 \vdash e_2 \Rightarrow v, M_2}{\sigma, M \vdash e_1.x := e_2 \Rightarrow v, M_2[a \mapsto v]}$$