CS320 Programming Languages
Homework #6: SRBFAE

Due: 06 November 2019

The goal of Homework #6 is to implement the interpreter of SRBFAE, which is the extended version of BFAE with general Sequences and Records.

1 Improving Sequences

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

```
<SRBFAE> ::= <number>
           | {+ <SRBFAE> <SRBFAE>}
           | {- <SRBFAE> <SRBFAE>}
           | <id>
           | {fun {<id>} <SRBFAE>}
           | {<SRBFAE> <SRBFAE>}
           | {newbox <SRBFAE>}
           | {setbox <SRSRBFAE> <SRSRBFAE>}
           | {openbox <SRSRBFAE>}
           | {seqn <SRBFAE> <SRBFAE>*}
```

For example:

```
test(run("""{{fun {b} {seqn {setbox b {+ 2 {openbox b}}}
                            {setbox b {+ 3 {openbox b}}}
                            {setbox b {+ 4 {openbox b}}}
                            {openbox b}}}
             {newbox 1}}"""), "10")
```

2 Records

Add `rec` and `get` forms for records, and also add `set` form that modifies the value of a record field:

```
<SRBFAE>  ::= ...
            | {rec {<id> <SRBFAE>}*}
            | {get <SRBFAE> <id>}
            | {set <SRBFAE> <id> <SRBFAE>}
```

A `rec` form allocates a new record. A `get` form accesses from the record produced by the sub-expression the value of the field named by the identifier. A `set` form changes within the record produced by the first sub-expression the value of the field named by the identifer; the value of the second sub-expression determines the field's new value, and that value is also the result of the set 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("{get {rec {x 1}} y}"), "no such field")
test(run("{{fun {r} {seqn {set r x 5} {get r x}}} {rec {x 1}}}"), "5")
```

You can implement a record in various ways, but using store is a possible option. Also define the `run` function which takes an expression as a string value and interprets it with an empty initial environment and empty initial store, and returns either a number string, `"function"`, `"box"`, `"record"` (without returning the store). Also, we will test your code with `"run"` function:

```
test(run("42"), "42")
test(run("{fun {x} x}"), "function")
test(run("{newbox 1}"), "box")
test(run("{rec}"), "record")
```