「プログラミング言語」課題

1029-24-9540 山崎啓太郎

April 22, 2013

1 Ex. 3.1

1.1 考え方

make-accumulator で返される関数内で、make-acumulator に与えられた引数に破壊的代入を使うことにより、それを更新していく。
3.1.scm では make-accumulator 関数の引数 x に対して、`(set! x (+ x add))` とすることで x を更新している。

1.2 実行例

test/3.1.scm より

```
Testing 3.1 ...
test (A 10), expects 15 ==> ok
test (A 10), expects 25 ==> ok
```

2 Ex. 3.3

2.1 考え方

make-account 関数で与えられたパスワードは、口座の操作ごとに入力する仕組みになっている。
3.3.scm では口座の操作の際に dispatch 関数が呼ばれているため、dispatch 関数内で口座の操作をする前にパスワードが正しいかどうかを判断し、パスワードが正しければ操作をし、正しくなければ”Incorrect password”を返せばよい。

2.2 実行例

test/3.3.scm より

```
Testing 3.3 ...
<Account test>---------------------------------------------------------------
test ((acc 'secret-password 'withdraw) 30), expects 70 ==> ok
test ((acc 'secret-password 'diposit) 40), expects 110 ==> ERROR: GOT #<<error> "Unkno
test ((acc 'secret-password 'withdraw) 300), expects #<error> ==> ERROR: GOT "Insuffic
test ((acc 'secret-password 'mes) 10), expects #<error> ==> ok
<Password test>--------------------------------------------------------------
test ((acc 'secret-password 'withdraw) 40), expects 60 ==> ok
test ((acc 'some-other-password 'diposit) 50), expects "Incorrect password" ==> ok
```