CSSE 304 Day 18 Summary

Final example

```
>(define f
     (lambda (x)
       (let ([a (lambda (y z) (+ x y z))])
         (lambda (b)
           (a (+ 5 b) x))))
   >((f 3) 4)
```

Now that we have seen how environments are supposed to work, we examine some possible implementation approaches

Environment ADT As usual for an ADT, we separate interface from representation and implementation.

Interface (mathematical model) An *environment* is a particular kind of finite function (from symbols to values):

a. **Interface:** If f is an environment, and s, s_1, …, s_k are symbols, then

(empty-env) ➔ ⌈∅⌉ (representation of the empty environment)

(apply-env ⌈f⌉ s) ➔ f(s) (get the value associated with s in the environment f)

(extend-env '(s_1 … s_k) (all of the s_i must be distinct, the v_i may be any Scheme values)
'(v_1 … v_k)
⌈f⌉) ➔ ⌈g⌉ where g(s) is v_i if $s=s_i$ (for some i, $1 \le i \le k$)
f(s) otherwise

One of the slides shows examples of the usage of this interface.

b. Possible environment representations in our interpreter

a. Scheme procedure

b. Record (via define-datatype)

c. List of lists (simple ribcage)

d. List of pairs (first part of each pair is a list, second part is a vector), a more efficient ribcage.

e. Alternate interface with succeed and fail callbacks (continuations).

2. Overview of the interpreter project. Details are on the slides. Use this space for any additional notes.