

CSSE 304 Day 31

1. A place for notes on the exam problems/solutions:

2. Convert the interpreter to CPS :

Add a new argument to eval-exp, the current continuation (a data-structure continuation)

```
(define eval-exp ;cps-version
  (lambda (exp env k)
    (cases expression exp ; look at typical cases
      [lit-exp (datum) (apply-k k datum)]
      [var-exp (id) (apply-env env id k fail-proc)]
      [lambda-exp (formals body)
        (apply-k k (closure formals body env))]
      [app-exp (rator rands)
        (eval-exp rator
                  env
                  (rator-k rands env k))]
      ...)))
```

```
(define-datatype continuation continuation?
  (test-k (then-exp expression?)
          (else-exp expression?)
          (env environment?)
          (k continuation?))
  (rator-k (rands (list-of? expression?))
           (env environment?)
           (k continuation?))
  (rands-k (proc-value scheme-value?)
           (k continuation?)) ; etc
```

3. **Convert to CPS: a clause of cases from eval-exp**

```
[let-exp (syms vals bodies)
  (let ([extended-env
        (extend-env syms
                     (map (lambda (x)
                           (eval-exp x env))
                          vals)
                     env)])
    (eval-bodies bodies extended-env))]
```

```
(define apply-k
  (lambda (k val)
    (cases continuation k
      [test-k (then-exp else-exp env k)
        (if val
            (eval-exp then-exp env k)
            (eval-exp else-exp env k))]
      [rator-k (rands env k)
        (eval-rands rands
                    env
                    (rands-k val k))]
      [rands-k (proc-value k)
        (apply-proc proc-value val k)])))
```

4. (define strange1
 (lambda (x)
 (display 1)
 (call/cc x)
 (display 2)))

(strange1 (call/cc (lambda (k) k)))