MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.001—Structure and Interpretation of Computer Programs
Spring 2006

Recitation 3
Recursion

Scheme

1. **Special Forms**
   (a) *define* - `(define (` *name arg1 arg2 ...*`)` *body*)
   Syntactis sugar for the following: `(define` *name* `(lambda (`*arg1 arg2 ...*`)` *body*`))`
   (b) *cond* - `(cond (`*test consequent*`) (`*test consequent*`) ... (else` *alternative*`))`
   Alternative to if when there are more than two cases. The value returned is the consequent where the first test evaluates to true (anything but #f). If no tests are true, evaluate and return the alternative, if any. The alternative `else` is optional. If a consequent is omitted, the value of the test is returned.

Problems

1. Consider the following definitions:

```
(define (our-display x)
        (display x)     ;this prints x to the screen
        x)              ;this returns x as the value

(define (count1 x)
   (cond ((= x 0) 0)
         (else (our-display x)
               (count1 (- x 1)))))

(define (count2 x)
   (cond ((= x 0) 0)
         (else (count2 (- x 1))
               (our-display x))))
```

   What will `(count1 4)` and `(count2 4)` display?

2. Write a procedure `fact` that computes the factorial of a number `n`.
   Plan:

3. Write a procedure `remainder` that computes the remainder of `num` divided by `divisor`.
   Plan:

4. Write a procedure that computes e.
   Plan:

5. Write an iterative procedure that computes e.
   Plan:

6. Write a procedure `fib` that computes the n^{th} fibonacci number.
   Plan:

7. Write a procedure that computes the golden ratio, ϕ.
   Plan:

8. Write a procedure that computes π.
   Plan: