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?

   count1: Display: 4321 return: 0
   count2: Display: 1234 return: 4

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

```
(define fact
  (lambda (n)
    (if (= n 0)
        1
        (* n (fact (- n 1))))))
```

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

```
(define remainder
  (lambda (num divisor)
    (if (> divisor num)
        num
        (remainder (- num divisor) divisor))))
```

4. Write a procedure that computes e.
   Plan:

$$e \approx \sum_{x=0}^{n} \frac{1}{x!}$$

```
(define (find-e n)
  (if (= n 0)
      1.0
      (+ (/ (fact n)) (find-e (- n 1)))))
```

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

```
(define (find-e n)
 (define (helper sum i)
   (if (= i 0)
      sum
      (helper (+ (/ (fact i)) sum) (- i 1))))
 (helper 1.0 n))
```

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

```
(define (fib n)
  (if (< n 2)
      n
      (+ (fib (- n 1)) (fib (- n 2)))))
```

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

$$\frac{a+b}{a} = \frac{a}{b} = \phi$$

```
(define (find-golden-ratio n)
  (/ (fib n) (fib (- n 1))))
```

8. Write a procedure that computes π.
   Plan:

$$\frac{\pi}{4} \approx \frac{1}{n}\sum_{i=1}^{n} g(x,y)$$

$$g(x,y) = \begin{cases} 1 \text{ if } x^2 + y^2 \leq 1 \\ 0 \text{ if } x^2 + y^2 > 1 \end{cases}$$

```
(define (guess-pi count)
  (* (/ (throw-n-darts count) count) 4))

(define (sum-of-squares x y) (+ (* x x) (* y y)))

(define (throw-n-darts n)
  (if (= n 0)
      0
      (if (< (sum-of-squares (random 1.0) (random 1.0)) 1)
  (+ 1 (throw-n-darts (- n 1)))
  (throw-n-darts (- n 1)))))
```