Recursion

Scheme

1. Special Forms

(a) \textit{define} - (\texttt{define ( name arg1 arg2 ... ) body})
Syntactic sugar for the following: (\texttt{define name (lambda (arg1 arg2 ...) body)})

(b) \textit{cond} - (\texttt{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 \texttt{else} is optional. If a consequent is omitted, the value of the test is returned.

Problems

1. Consider the following definitions:

\begin{verbatim}
(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))))
\end{verbatim}

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

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

2. Write a procedure \texttt{fact} that computes the factorial of a number \texttt{n}.

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

3. 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)))))

4. 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))

5. 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)))))

6. Write a procedure that computes the golden ratio, $\phi$.
   Plan:

   \[
   \frac{a+b}{a} = \frac{a}{b} = \phi
   \]

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