CS 326 – Project #10

Purpose: Become familiar with the functional language Scheme.
Points: 125

Assignment:

- Download and install the DrRacket (http://racket-lang.org/). Racket is a dialect of List and descendant of Scheme. DrRacket is the integrated development environment. *Note*, the provided template will specify the language.

- Become familiar with basic Scheme expression evaluation. Evaluate the following expressions in Scheme. (10 pts)

```
(+ 1 2)
(+ 1 2 3 4 5)
(+ (* 2 3) (* 4 5))
(+ (* 9 4) (/ 15 3))
(- (* (+ 11 8) 3 4) 5)
(* (- (+ (+ 4 7) 2) 3) 2)
(+ 1 (/ (- 2 3) (* 4 5)))
(+ 1 0.5)

(define (f x)
   (+ (* x x) (* x x x) ) )
(f 3)
(f 5)

(define (square n)
     (* n n) )
(square 4)
(square 8)

(< 4 2)
(> (+ 5 7) (/ 141 114))
```

For the logicals, the answer will be true or false.

- Translate the following algebraic formulas into Scheme's notation and submit the result for each. Type the translation into Scheme's interaction window for checking. (10 pts)

```
7 – (4 - 5)
(8 * 7) - (10 + 5)
(2 * 3) + (4 * 5)
2 + 3 + 4
5 * (4 + (-5 - -3))
(36 / 6) / 2
3 / (5 - (1 / 7))
2^3 + 3^2
```

For example, translating 7 - (4 - 5) into Scheme's notation yields (- 7 (- 4 5)).

- Become familiar with simple Scheme functions.
  (30 pts, #1-#4, 5 pts each, #5 10 pts)

  cube takes a number x and returns x^3

  rarea function that computes the area of a circle given its radius

  poly1 takes a number x and returns $f(x) = 5x^2 + 12x + 36$

  poly2 takes a number x and returns $f(x) = 2x^3 + 4x^2 + 7x + 17$

  payment computes payment: $amount * \frac{(irate * (1 + irate)^{term})}{((1 + irate)^{term} - 1.0)}$

- Become familiar with simple Scheme recursive functions. (50 pts, 10 pts each)

  fact - takes a number x, and returns $x!$ (x factorial). Must be recursive.

```
fact(x) =          if (x=0)
                         fact = 1
                   else
                         fact = ( fact(n-1) * n )
```

  fib - takes a number x and returns the Fibonacci number

```
fib(x) =           if (x=0)
                         fib = 0
                   if (x=1)
                         fib = 1
                   else
                         fib = ( fib(n-1) + fib(n-2) )
```

  harmonic - takes a number ***x*** the harmonic number.

```
harmonic(x) =      if (x=0)
                         harmonic = 1
                   else
                         harmonic = ( harmonic(x-1) + (1.0/x) )
```

  tak - takes three numbers and computes the ***tak*** number.

```
tak(x,y,z) =       if (y >= x)
                         return(z);
                   else
                         return( tak( tak(x-1, y, z),
                                  tak(y-1, z, x), tak(z-1, x, y)))
```

  exponent – calculates x^y

```
exponent(x y) =    if (y=0)
                         exponent = 1
                   else
                         exponent = x * exponent(x y-1)
```

- Become familiar with basic lists. Evaluate the following expressions in Scheme. (25 pts)

```
(list '2 '3 '4 '5 '6)
(list 'a 'b 'c 'd 'e)
(list 'one 'two 'three 'four)
(list '2 '(3 4) '5)
(list '1 '2 'fred '4 '5)
(cons 2 '(3 4))
(cons 3 '(5 7 (9 11)))
(car '(2 4 6 8))
(car '((2 5) 9 13))
(cdr '(5 7 9 11))
(cdr '(122 123))
(length '(11 13 15 17 19))
(reverse '(71 73 75 77 79))
(append '(12 13) '(15 17 19))
```

A set of test calls for the functions will be provided.
All expressions and functions should be in a single source file.

Submission:

1) Submit a copy of the Scheme program (definitions window).
2) Submit a copy of the results (interactions window).
   To print the interaction window, use "File → Print Interactions"