SCHEME PROGRAMMING!!!!!!

Variables and Types

1. Variable names can be upper or lower case and only 1st character needs to be 'char'
2. Types

#t	->	boolean true
#f	->	boolean false
		numbers
"hello there"		Strings
#(6 7 8)		-vectors (fixed length tuples)
(a b c), ( )		lists
'abc		symbols (unevaluated expression)
(a . b)		dotted pairs (result of using (cons a b))

EVALUATION
Scheme tries to evaluate anything that has not been quoted.
If the thing is sa variable, it tries to evaluate what the variable stood for.

Ex

- 2
  2 (evaluating a number just evaluates to itself)
- '(a b)
  '(a b)
- (+ 2 3)
  5
- (= 2 3)
  #f
- (+ 2 (- 3 4))
  1

Generally to evaluate a list scheme assumes list means
(operation expr1 expr2 … exprn)
Computes the expressions first, then applies the operation to get the value.

- (cons 7 ( ))
  (7)
- (cons 7 (6)) (cons concatenates the list)
  (7 6)
- (car (7 6)) (car retrieves first item in the list)
  7
- (cdr (7 6 5)) (returns everything AFTER the first element in the list)
  (6 5)
- (cons 7 6) (creates a dotted pair)
  (7 . 6)

Making definitions

General format (define var_name value)

- (define y 10) sends the value 10 into variable y
- (define z 'hello) sends the SYMBOL hello into z
- (define mult

```
        (lambda (x y)
                (* x y) ) )
```

- (mult 5 6)
  30
- (define x 7)
- (set! X 6) (changes the value of X to 6)
- (display x)
- (newline) (writes a \n to screen)
- (begin
  ```
        (display “hello”)
        (display “ there”) )
  ```
  hello there
- (let ((x 5) (y 4) (* x y))
  20
-