1. Arithmetic operations:

> (+ 3 4)

7

> (\* 5 6)

30

> (/ 50 10)

5

> (min 44 33 88 99 11)

11

> (max 100 99 45 33 1000)

1000

>

1. Compute values:

The value of (sin (π/2) + cos( 0))

> (+ (cos 0) (sin(/ pi 2)))

#i2.0

>

1. Defining functions:

Lambda expression is a nameless function and includes the word lambda and it is called lambda function:

> ((lambda (x) (\* x x))7)

49

> ((lambda (x) (+ x x))7)

14

>

Scheme special function define serves two fundamental needs:

* 1. Bind name to a value:

> (define x 4)

> x

4

> (define y 5)

> y

5

>

* 1. Bind name to a lambda expression:

(define (square x) (\* x x))

> (square 5)

25

> (square 100)

10000

>

(define (twice x) (+ x x))

> (twice 5)

10

> (twice 7)

14

>

1. Conditional and recursive structure:

Factorial:

;Facotrial

(define (factorial n)

( if (= n 1)

1

(\* n (factorial(- n 1)))

)

)

> (factorial 5)

120

> (factorial 50)

30414093201713378043612608166064768844377641568960512000000000000

>

Fibonacci:

;Fibonacci

(define (fibonacci n)

(if (or (= n 1)(= n 0))

n

(+ (fibonacci (- n 1)) (fibonacci (- n 2)))

)

)

> (fibonacci 6)

8

>

1. Lists quote

> (quote (1 2 3 4 5))

(list 1 2 3 4 5)

> '(1 2 3 4 5)

(list 1 2 3 4 5)

>

1. Lists cons and list:

> > (cons 1 (cons 2 (cons 3 (cons 4 (cons 5 null)))))

>

(list 1 2 3 4 5)

> (list 1 2 3 4 5)

(list 1 2 3 4 5)

>

1. List car, cdr

> (car '(1 2 3 4 5))

1

> (cdr '(1 2 3 4 5))

(list 2 3 4 5)

> (car (cdr '(1 2 3 4 5)))

2

> (cdr(cdr '(1 2 3 4 5)))

(list 3 4 5)

> (cdr(cdr(cdr'(1 2 3 4 5))))

(list 4 5)

> (car(cdr(cdr(cdr '(1 2 3 4 5 6)))))

4

>