* 1. (\* (+ 22 42) (\* 54 99))
  2. (\* (\* (+ 22 42) 54) 99)
  3. (\* (+ (+ 64 102) 16) (/ 44 22))
  4. Although a and b return with same value they are quite in the process mathematically and in terms of scheme. A takes the product of two arguments that must be evaluated first meaning first scheme must take the addition of 22 and 42 and then the multiple of 54 and 99. However, B works a little different where it takes the addition of 22 and 42 but then multiply it with 54. Then the result is multiplied with 99. It is done in the same method in math and scheme replicates that.
  5. Yes, the rule of precedence is important and needed to write operations in scheme because scheme will evaluate and apply starting from left to right hence no precedence. This means that in case of 3+4\*5. We must explicitly state in scheme that we need 4 multiplied by 5 first and that sum that with 3. Otherwise we could write a operation that takes the sum of 3 and 4 first which is wrong.

1. 1. [99] [54] x [42] [22] + x
   2. [99] [54] [42] [22] + x x
   3. [64] [102] \* [16] [44] [22] / \*+
2. 1. (define (inc x) (+ x 1))

(define (inc2 x) (inc x))

* 1. (define (square x) (\* x x))

(define (square2 x) (square x))

* 1. (define (square x) (\* x x))

(define (forth x) (square (square x)))

(define (quad1 x) (+(square x) 1))

(define (quad2 x) (+(\* 16 (forth x)) 22))

(define (p x) (\*(forth (quad1 x)) (square (quad2 x))))

* 1. (define (square x) (\* x x))

(define (forth x) (square (square x)))

(define (sixteenth x) (fourth (fourth x)))

(define (sixty-fourth x) (sixteenth (fourth x)))

* 1. Yes, without the use of previous function to create sixty-fourth, scheme would needed multiplication operation 64 times in order to get power of 64.

1. 1. (define (square x) (\* x x))

(define (radican sig) (\*(\* 2 sig) 3.142))

(define (N x sig) (\*(/ 1 (sqrt (radican sig)) (expt 2.718 (/ (square x) (\* -2 sig))))))

* 1. (define (power theta) (\* theta (/ 2 3.142)))

(define (fsprial theta) (expt 1.618 (power theta)))

* 1. (define (malth t p a) (\* p (expt 2 (\* a t))))
  2. (define (num Pi Ps) (\* Ps Pi))

(define (denom Pi Ps alpha t) (+ Pi (\*(- Ps Pi) (expt 2.718 (\* -1 (\* alpha t))))))

(define (singlespecies Pi Ps alpha t) (/(num Pi Ps) (denom Pi Ps alpha t)))