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

(define (number-square x)

(if (= x 0)

0

(+ (square x) (number-square (- x 1))))

* 1. (define (num-even x)

(if (= x 0)

0

(+(\* 2 x)(num-even (- x 1)))))

1. (define (recursiveK k)

(if (= k 2)

0.5

(\*(- 1 (/ 1 k))(recursiveK (- k 1)))))

1. (define ( divides a b ) (= 0 ( modulo b a )))

(define (divisors-upto n k)

(cond ((= k 0) 0)

((= n 0) 0)

((= k 1) 1)

((divides k n) (+ 1 (divisors-upto n (- k 1))))

(else (divisors-upto n (- k 1)))

)

)

1. (define (divisors n) (divisors-upto n n))
2. (define (series k)

(if (= k 0)

0

(\*(expt -1 (- k 1) ) (/ 4 (-(\* 2 k) 1)))

)

)

(define (sum-series k)(+(series k)(series (- k 1))))

1. 1. It computed expt 300 times to get a value of either 1 or -1 each time.
   2. (define (alternating-sign k)

(if (= 0 (remainder k 2))

-1

1

)

)

(define (new-series k)

(if (= k 0)

0

(\* (alternating-sign k)(/ 4 (-(\* 2 k) 1)))

)

)

(define (sum-series k)(+(series k)(series (- k 1))))

1. New-if and if are not different and the same where they evaluate a condition and apply a clause if said condition is true else apply another clause in other situations. Both work the same and have no differences
   1. (define (factorial n)
   2. (if (= n 0)

1

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

)

)

(define (new-sin x n)(\*(expt -1 n)(/(expt x (+(\* 2 n)1)) (factorial(+(\* 2 n)1)))))