1. Convert and Evaluate the following arithmetic expression to scheme expression: [CO01]
2. Rewrite the following procedure using a ***cond*** instead of the ***ifs***: [CO01]

(define (sign number)

(if (< number 0)

'negative

(if (= number 0)

'zero

'positive)))

* 1. What will be the result of the call (trick (/ 1 0) 5) in normal and applicative order of evaluation, given the definition of trick as (define (trick x y) (\* y y)). Justify your result.
  2. How many times does **‘+’** get called in evaluating the call ( inc ( inc ( + 3 2 ) ) ) where inc is defined as (define (inc x) (+ x 1)) using applicative and normal orders of evaluation? Justify the result.

1. Write a procedure (ab+c a b c) which accepts a, b, c as arguments and returns (a\*b) + c without using the \* operator.
2. Find the ***order of growth in terms of steps*** for the following procedure and explain your answer.

(define (mystery x)

(cond((= x 0) 0)

((= x 1) 1)

(else (+ (mystery (- x 1))

(mystery (/ x x)) ))))

1. Find the ***order of growth in terms of space*** for the following procedure and explain your answer.

(define (foo n)

(if (= (remainder n 7) 0)

n

(foo (- n 1))))

1. Rewrite the below scheme procedure which generates a recursive process in calculating the log of a number into a procedure that generates an iterative process. [CO01]

(define (lg n)

(if (= n 1)

0

(+ (lg (floor (/ n 2))) 1)))

1. Determine and justify the order of growth in terms of time for the following functions using Θ notation. (Hint: you will only need to use one or more of the following for your answers: Θ(1), Θ(log n), Θ(n), Θ(n2) and Θ(2n).) [5] [CO01]
2. **(define (square n)**

**(cond ((= n 0) 0)**

**((even? n) (\* (square (quotient n 2)) 4))**

**(else (+ (square (- n 1)) (- (+ n n) 1))) ) )**

1. **(define (func x)**

**(func2 (\* 2 x)))**

**(define (func2 x)**

**(if (< x 1) 0**

**(+ (factorial x) (func2 (/ x 2)))))**

**(define (factorial n)**

**(if (= n 0) 1 (\* n (factorial (- n 1)))))**

1. **(define (mul n)**

**(\* pi (sqrt 4) n))**

1. Define a scheme procedure that takes a natural number n as input and tests the Collatz conjecture for n. A Collatz conjecture takes any positive integer as input and each term is obtained from the previous term as follows: if the previous term is even, the next term is half of the previous term. If the previous term is odd, the next term is 3 times the previous term plus 1. The conjecture is that no matter what value of *n*, the sequence will always reach 1.
2. Classify the following processes as iterative or recursive or neither with justification in each case: [CO01]
3. **(define foo (lambda (x) (\* x 10)))**
4. **(define (quotient a b)**

**(if (< a b) 0**

**(+ 1 (quotient (- a b) b))))**

1. **(define (quotient a b)**

**(define (quotient-helper a b answer)**

**(if (< a b) answer**

**(quotient-helper (- a b) b (+ 1 answer))))**

**(quotient-helper a b 0))**