Problem 1

Part A:

In unary representation the numbers represented as a list of #ts. The length of the list is the number.

In BigNum representation the number represents as a list of coefficients of an arbitrary number N’s exponentials. For example if N is 4, the first three element of the list are coefficients of 40, 41, 42 respectively.

Part B:

Unary:

> (define (create n) (if (zero? n) '() (cons #t (create (- n 1)))))

> (define (is-zero? lst) (if (null? lst) #t #f))

> (define (predecessor lst) (if (null? lst) “error” (cdr lst)))

BigNum:

> (define (create n i) (if (zero? n) '() (cons (modulo n i) (create (quotient n i) i))))

> (define (is-zero? lst) (if (null? lst) #t #f))

> (define (predecessor lst i) (if (null? lst) "error" (if (= 0 (car lst)) (if (null? (cdr lst)) "error" (cons (- i 1) (predecessor (cdr lst)))) (cons (- (car lst) 1) (cdr lst)))))

Problem 2

Part A:

> (define (count-free-occurrences var exp) (occurrences-free var exp 0))

> (define (occurrences-free var exp i) (cond((symbol? exp) (if (equal? var exp) (+ 1 i) i)) ((equal? (car exp) 'lambda) (if (equal? var (car (cadr exp))) i (+ i (occurrences-free var (caddr exp) 0)))) (else (+ i (+ (occurrences-free var (car exp) 0) (occurrences-free var (cadr exp) 0))))))