(define zero

(lambda ()

'()))

(define is-zero?

(lambda (lst)

(null? lst)))

(define convert

(lambda (num)

(cond

(( < num 10) num)

((eqv? num 10) 'A)

((eqv? num 11) 'B)

((eqv? num 12) 'C)

((eqv? num 13) 'D)

((eqv? num 14) 'E)

((eqv? num 15) 'F)

(else num))))

(define devert

(lambda (num)

(cond

((eqv? num 'A) 10)

((eqv? num 'B) 11)

((eqv? num 'C) 12)

((eqv? num 'D) 13)

((eqv? num 'E) 14)

((eqv? num 'F) 15)

(else num))))

(define success-helper

(lambda (num base)

(if (is-zero? (quotient num base))

(list (remainder num base))

(cons (remainder num base) (success-helper (quotient num base) base)))))

(define N 16)

(define success

(lambda (lst)

(cond

((is-zero? lst) '(1))

((< (devert (car lst)) (- N 1)) (cons (convert (+ (devert (car lst)) 1)) (cdr lst)))

(else (cons 0 (success (cdr lst)))))))

(define list-zero

(lambda (lst)

(if (null? lst)

'()

(if (zero? (car lst))

(list-zero (cdr lst))

#f))))

(define precess

(lambda (lst)

(if (eqv? (list-zero lst) '())

lst

(cond

((is-zero? lst) '())

((> (devert (car lst)) 0) (cons (convert (- (devert (car lst)) 1)) (cdr lst)))

(else (cons 'F (precess (cdr lst))))))))