```
#lang racket


(define (repeat-i l n i)
  (if (empty? l)
      -1
      (if (= (car l) n)
          i
          (repeat-i (cdr l) n (+ i 1)))))

(define (repeat-count l n)
  (+ (repeat-i l n 0) 1))


(define (cycle-h hist denom)
  (define next (modulo (* (car hist) 10) denom))
  (define cycle_len (repeat-count hist next))
  (if (= cycle_len 0)
      (cycle-h (cons next hist) denom)
      cycle_len))

(define (cycle denom)
  (cycle-h (list 1) denom))

(define (sol-h i max roof)
  (if (>= i roof)
      max
      (if (> (cycle i) (cycle max))
          (sol-h (+ i 1) i roof)
          (sol-h (+ i 1) max roof))))

(define (sol)
  (sol-h 1 1 1000))

(sol)
```