```racket
#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)
```

```

; plan - this should be obvious
; but it doesn't really make sense
; how am I supposed to tell when it's repeating??
; let's see. anything that isn't evey going to go into a multiple of 10
; so that means:
; if it's made up of literally anything other than 2s and 5s
; and if that's the case... then it's repeating
; but how do I know when it starts? and when it ends?
; I don't really want to be doing guesswork
; after all these are mathematical problems
; so then... what am I supposed to do???
; let's think.........................
; ok if i were to brute force this
; it would go something like this ig:
; keep a running list of the digits that come up
; and if a digit comes up that's been there before
; ... argh this is sooo messy
; so, other ways:
; wait i got it (sort of)
; it's until the remainder is the same!
; so like you divide 10 by 3
; and then you get a remainder of 1
; which is 10 / 10!
; and with 1 / 7
; you start with 10 / 7
; and you get 1
; and are left with 3
; and then 30 / 7
; which gets you 4
; and leaves 2
; and then 20 / 7
; which gets 2
; leaving 6
; so then 60 / 7
; which gets 8
; and leaves 4
; 40 / 7
; gets 5
; and leaves 5
; 50 / 7
; gets 7
;
```