```
;;; Partiel L1-Info Janvier 2014

;;; Q1. Sur 1.5 pts (0.5 + 0.5 + 0.5)
; E : (append L1 (append L2 L3)) consomme n1 + n2 cons
; F : (append (append L1 L2) L3) consomme 2*n1 + n2 cons, donc E est plus efficace !

;;; Q2. Sur 1 pt
; une fonction recursive est iterative si l'appel recursif est terminal [pas d'enveloppe].

;;; Q3. Sur 1.5 pts (0.5 + 0.5 + 0.5)
; E = +     F = (lambda (x) (* 2 x))       G = number?

;;; Q4. Sur 1.5 pts au prorata
(define (tirage)
  (= 1 (gcd (+ 1 (random 100000000)) (+ 1 (random 100000000)))))

;;; Q5. Sur 3 pts au prorata (2 + 0.5 + 0.5)
(define (element i L)    ; on suppose i < length(L)
  (if (= i 0)
      (first L)
      (element (- i 1) (rest L))))   ; ITERATIVE ! Complexite en O(i) et independante de n

;;; Q6. Sur 3 pts au prorata
(define (proba N)
  (local [(define (iter N acc)   ; acc compte le nombre de succes
            (cond ((zero? N) acc)
                  ((tirage) (iter (- N 1) (+ acc 1)))
                  (else (iter (- N 1) acc))))]
    (exact->inexact (/ (iter N 0) N))))

;(show (proba 10000))

;;; Q7. Sur 2 pts au prorata
(define (inter E F)       ; rec. sur E
  (cond ((empty? E) empty)
        ((member (first E) F) (cons (first E) (inter (rest E) F)))
        (else (inter (rest E) F))))

;(show (inter '(re sol mi fa) '(si la fa sol)))

;;; Q8. Sur 2 pts au prorata
(define (subst B1 B2 A)  ; dans l'arbre A, on remplace chaque sous-arbre B1 par B2
  (cond ((equal? A B1) B2)
        ((feuille? A) A)
        (else (arbre (racine A) (subst B1 B2 (fg A)) (subst B1 B2 (fd A))))))

;(show (subst '(+ x 1) 'u '(+ (* (+ x 1) x) (+ x 1))))

;;; Q9. Sur 6 pts (0.5 + 0.5 + 1 + 1 + 1 + 1 + 1)
(define (animation)
  (local [(define FOND (underlay (rectangle 300 300 'solid "yellow") (line 300 0 "black")))
          (define IMG (circle 10 'solid "red"))
          (define-struct bille (x y dx dy))
          ; Le monde est une liste de 30 billes espac√©es de 10 en 10 sur la droite du milieu AB
          (define INIT
            (build-list 30 (lambda (i) (make-bille (random 300) 150 (- (random 5) 2) (- (random 11))))))
          (define (deplacer b)
            (make-bille (+ (bille-x b) (bille-dx b)) (+ (bille-y b) (bille-dy b))
                        (bille-dx b) (+ (bille-dy b) 0.5)))
          (define (suivant L)
            (map deplacer L))
          (define (dessiner L)
            (if (empty? L)
                FOND
                (local [(define b (first L))]
                  (place-image IMG (bille-x b) (bille-y b) (dessiner (rest L))))))
          (define (final? L)
            (andmap (lambda (b) (> (bille-y b) 310)) L))]
    (big-bang INIT
              (on-tick suivant)
              (on-draw dessiner)
              (stop-when final?)
              (name "Bon No√´l !"))))

;(animation)
```