CONTROLE FLASH Numéro 1 (PF1) *45 minutes*

DATE : mercredi, à 11h30 **SECTION : Groupe 2, L1-Info**

NOM : **PRENOM :**

Question 1a : Programmez une fonction `(drop L n)` retournant la liste `L` privée de ses `n` premiers éléments. On suppose n ≤ length(L). Ex: `(drop '(a b c d e) 3)` → `(d e)`

```
(define (drop L n)
```

Question 1b : La fonction `drop` rédigée ci-dessus est-elle **itérative** ? A quoi le voyez-vous à la simple lecture du texte de cette fonction ?

Question 2a : On considère la fonction `(polygone n c)` ci-dessous, qui construit une liste de n ***points*** (structures de type `posn`) tirés au hasard avec des coordonnées dans [0,c[.

```
(define (polygone n c)
  (if (= n 0)
      empty
      (cons (make-posn (random c) (random c))
            (polygone (- n 1) c))))
```

```
> (polygone 5 300)      ; par exemple
(#(struct:posn 224 239) #(struct:posn 52 169) #(struct:posn 12 220)
 #(struct:posn 277 208) #(struct:posn 65 225))
```

A quoi voyez-vous textuellement que cette fonction n'est **pas itérative** ?

Question 2b : Reprogrammez la fonction `(polygone n c)` de manière **itérative**.

Question 2c : Comment reprogrammeriez-vous **en une ligne**, sans récurrence ni itération, la fonction `(polygone n c)` ?

Question 3 : Programmez une fonction `(rmap f pred? L)` fonctionnant comme `(map f L)` mais ne conservant que les éléments du résultat qui vérifient le prédicat `pred?`. Vous utiliserez le style de programmation de votre choix. Exemple :

```
(rmap sqr (lambda (x) (> x 20)) '(3 -6 5 1 10 2)) → (36 25 100)
```

Question 4 : **Système de particules**. L'animation d'une pluie se déroule dans un canvas carré bleu clair (*lightblue*) de taille 200. **Le monde sera une liste L de gouttes** (le nombre de gouttes n'est pas constant). Chaque goutte - que l'on représentera par une structure - va naître à une position x,y aléatoire du canvas et va donner naissance à un cercle bleu pur (*blue*) dont le rayon `r` qui est nul à la naissance grandit d'un pixel à chaque top d'horloge. Une goutte meurt lorsque le rayon de son cercle atteint la valeur 4. A chaque top d'horloge deux nouvelles gouttes vont naître et d'autres vont mourir. Au début de l'animation il ne pleut pas ! En utilisant la fonction Racket `(current-milliseconds)` qui retourne l'heure courante sous la forme d'un entier exprimé en *millisecondes*, arrangez-vous pour que l'animation dure 15 secondes, quelle que soit la vitesse d'horloge.

```
(define FOND
(define-struct goutte
(define INIT
```

```
(define (suivant L)                          ; monde → monde
```

```
(define (dessiner L)                         ; monde → scène
```

```
; l'animation doit stopper au bout de 15 sec. Votre code :
```

; Laissez tomber le big-bang, on s'en occupe :-)