**CM 1**

On utilise docteur racket

Lisp fortement typé, d'autres langages de programmation en fonctionnel comme CAML par exemple, mais lui il est pas typé

Pas de séparation entre le programme et les données, dans le langage Skim, les données sont souvent dans des listes

On utilise essentiellement de la récursivité super

Une liste du coup c'est de la forme liste = () si c'est vide. Ou alors : (T|Q), Tête et Queue qui forment une liste

Q est une liste, T est quelque chose

Si on prend l'exemple de la liste : (1, 2, 3), c'est égal à ((1|(2, 3)), (2, 3) = (2|(3)), (3) = (3|()) donc c'est bien une liste car c'est bien une tête et une queue, par la récursivité on voit à la fin sa structure

Les constantes du langage :

* Des String mais qui doivent commencer forcément par une lettre.
* Les nombres : suite de chiffres ou suite de chiffres . suite de chiffres. On peut les écrire en notation ingénieur, genre 5e -8
* Les booléens : vrai ou faux (#T ou #F)

Notation préfixée, on met la fonction puis les paramètres :

x + y -> +x y -> (+x y)

On peut faire des enchainements grâce à define :

(define longueur (+3 5)) par exemple

On peut faire des lambdas (c’est trop fort) :

(define factorielle (lambda (n) (if (= n 0)

1

(\* n (factorielle (- n 1)))))

Et en fait ça c’est une fonction classique du coup.

Pour empêcher l’évaluation d’une liste : on utilise la quote.

(define L (quote (1 2 3))) quote (1 2 3) va renvoyer 1 2 3

On utilise sinon la quote ‘

On peut mettre autant d’arguments qu’on veut : (+ 1 2 3 4 5 6 7 8 9)

Pour accéder à la tête de la liste : c’est car

Pour accéder à la queue de la liste : c’est cdr

En fait quand on met (1 2 3) il essaye de l’évaluer et de faire une addition par exemple. Il faut utiliser la quote pour ça.

Si on fait ?(car (1 2 3)) ça plante

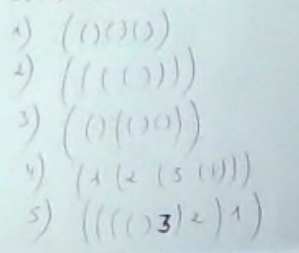
?(car ‘(1 2 3)) ça donne 1

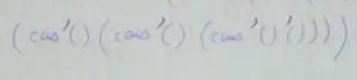
CDR = liste – la tête

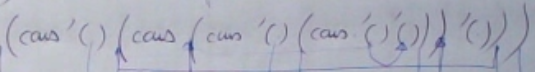
Si on veut récupérer le 3 :

?(car (cdr (cdr ‘(1 2 3))))

(cons 1 ‘(2 3)) va donner la liste (1 2 3)





1. (cons () ‘(cons ‘() ‘())))
2. (cons(cons(cons'()'())'())'())
3. 
4. 