

Le langage Racket

- Un langage de programmation fonctionnelle
 - 13/11/2021

1. Exercices

Question 2

Définir une fonction **f** telle que $f(x) = \sum_{i=1}^x i^2$

Appeler (**f** 2) retourne 5.

Appeler (**f** 10) retourne 385.

(*fonctions utiles : =, +, *, -*)

1. Exercices

Solution 2

```
1 (define (f x)
2   (if (= x 0)
3       0
4       (+ (* x x) (f (- x 1))))
5   ))
6 (f 1)
7 (f 10)
```

```
5
385
```

1. Exercices

Question 3

Définir une fonction `f` telle que $f(x, y) = \sum_{i=x}^y i$.

Appeler `(f 1 2)` retourne 3.

Appeler `(f 1 10)` retourne 55.

(fonctions utiles : `=, +, -`)

Solution 3

Première solution³⁰ :

```
1 (define (f x y) (if (= x y) y (+ y (f x (- y 1)))))  
2 (f 1 2)  
3 (f 1 10)
```

```
3  
55
```

Deuxième solution :

```
1 (define (f x y) (if (= x y) x (+ x (f (+ x 1) y))))  
2 (f 1 2)  
3 (f 1 10)
```

```
3  
55
```

30. Dans laquelle ... ($= x y$) y ... peut également s'écrire ... ($= x y$) x ... ou ... ($= y 0$) 0 ... ou ... ($= y 0$) y ... ou ... ($= y 1$) 1 ... ou ... ($= y 1$) y ...

1. Exercices

Question 4

Définir une fonction **f** telle que $f(x, y) = \sum_{i=x}^y i$ avec $x > 0$ et i pair.

Appeler (**f** 1 2) retourne 2.

Appeler (**f** 1 10) retourne 30.

(*fonctions utiles : =, <, >, even?, +, -, modulo*)

1. Exercices

Première solution :

```
1 (define (f x y)
2   (if (= x y)
3     (if (even? y) y 0)
4     (+ (if (even? y) y 0)
5       (f x (- y 1))))
6   ))
7 (f 1 2)
8 (f 1 10)
```

2

30

1. Exercices

```
1 (define (f x y)
2   (if (even? x)
3     (if (> x y) 0 (+ x (f (+ 2 x) y)))
4     (f (+ x 1) y)))
5 (f 1 2)
6 (f 1 10)
```

```
2
30
```

1. Exercices

```
1 (define (f x y)
2   (if (> x y)
3     0
4     (+ (* x (- 1 (modulo x 2))) (f (+ x 1) y)))
5   )
6 (f 1 2)
7 (f 1 10)
```

```
2
30
```

1. Exercices

Question 5

Définir une fonction `f` telle que $f(x, y) = x^y$ avec $y \geq 0$.

Appeler (`f 1 2`) retourne 1.

Appeler (`f 2 8`) retourne 256.

Appeler (`f 8 16`) retourne 281474976710656.

(*fonctions utiles*²⁹ : `=`, `*`, `-`)

1. Exercices

Solution 5

```
1 (define (f x y) (if (= y 0) 1 (* x (f x (- y 1)))))  
2 (f 1 2)  
3 (f 2 8)  
4 (f 8 16)
```

```
1  
256  
281474976710656
```

1. Exercices

Question 6

Définir la fonction factorielle `f` telle que $f(x) = \prod_{i=1}^x i$.

(Pour toute valeur de x inférieure ou égale à 1, $f(x)$ retourne 1).

Appeler `(f -1)` retourne 1.

Appeler `(f 1)` retourne 1.

Appeler `(f 8)` retourne 40320.

1. Exercices

Solution 6

Première solution (réursive non-terminale, ici réursive enveloppée) :

```
1 (define (f x) (if (<= x 1) 1 (* x (f (- x 1)))))  
2 (f -1)  
3 (f 1)  
4 (f 8)  
5 (f 16)
```

```
1  
1  
40320  
20922789888000
```

1. Exercices

Deuxième solution (réursive terminale) :

```
1 (define (f x) (myf x 1))
2 (define (myf x acc) (if (<= x 1) acc (myf (- x 1) (* acc x))))
3 (f -1)
4 (f 1)
5 (f 8)
6 (f 16)
```

```
1
1
40320
20922789888000
```

1. Exercices

Troisième solution (réursive terminale avec encapsulation) :

```
1 (define (f x)
2   (define (myf x acc)
3     (if (<= x 1) acc (myf (- x 1) (* acc x))))
4   (myf x 1))
5 (f -1)
6 (f 1)
7 (f 8)
8 (f 16)
```

```
1
1
40320
20922789888000
```

1. Exercices

Question 7

Définir la fonction de Fibonacci $\mathcal{F}(x)$ telle que

$$\begin{cases} \mathcal{F}(0) = 0 \\ \mathcal{F}(1) = 1 \\ \mathcal{F}(n + 2) = \mathcal{F}(n + 1) + \mathcal{F}(n) \end{cases}$$

Appeler (F 8) retourne 21.

Appeler (F 16) retourne 987.

(fonctions utiles : `<=`, `=`, `+`, `-`)

1. Exercices

Solution 7

```
1 (define (f x)
2   (if (<= x 0) 0
3     (if (= x 1) 1
4       (+ (f (- x 1)) (f (- x 2))))
5     )))
6 (f 8)
7 (f 16)
```

```
21
987
```

1. Exercices

Solution 7 _____

```
1 (define (f x)
2   (define (g x a b)
3     (if (<= x 0) a
4         (g (- x 1) b (+ a b)))
5   )
6   (g x 0 1))
7 (f 8)
8 (f 16)
```

21
987

1. Exercices

Question 12

Définir une fonction pgcd selon la méthode des différences.

Principe : un nombre est un diviseur de 2 nombres a et b s'il est un diviseur de la différence entre a et b.

Exemple pour 21 et 28 : $28 - 21 = 7$; $21 - 7 = 14$; $14 - 7 = 7$; $7 - 7 = 0$; donc 7 est pgcd de 21 et 28.

Exemple pour 8 et 13 : $13 - 8 = 5$; $8 - 5 = 3$; $5 - 3 = 2$; $3 - 2 = 1$; $2 - 1 = 1$; $1 - 1 = 0$; donc 1 est pgcd de 8 et 13.

Appeler (pgcd 21 28) retourne 7.

Appeler (pgcd 8 13) retourne 1.

(fonctions utiles : =, <, -)

1. Exercices

Solution 12

```
1 (define (pgcd a b)
2   (if (= a b) a
3       (if (< a b) (pgcd a (- b a)) (pgcd b (- a b)))))
4 (pgcd 21 28)
5 (pgcd 8 13)
```

```
7
1
```

Première solution :

```
1 (define (L->2L L)
2   (if (empty? L) empty
3       (cons (first L) (cons (first L) (L->2L (rest L)))))
4   )))
5 (L->2L '(1 2 3 4 5))
```

```
'(1 1 2 2 3 3 4 4 5 5)
```

Deuxième solution :

```
1 (define (L->2L L)
2   (if (empty? L) empty
3       (append (list (first L) (first L)) (L->2L (rest L))))
4   )))
5 (L->2L '(1 2 3 4 5))
```

```
'(1 1 2 2 3 3 4 4 5 5)
```

Troisième solution (modification de la première avec `let`) :

```
1 (define (L->2L L)
2   (if (empty? L) empty
3     (let ([i (first L)])
4       (cons i (cons i (L->2L (rest L)))))
5     )))
6 (L->2L '(1 2 3 4 5))
```

```
'(1 1 2 2 3 3 4 4 5 5)
```

Quatrième solution (avec encapsulation et `map`) :

```
1 (define (L->2L L)
2   (define (f e) (list e e))
3   (flatten (map f L)))
4 )
5 (L->2L (list 1 2 3))
```

```
'(1 1 2 2 3 3 4 4 5 5)
```