ENSEIRB-Matmeca
2019-2020

1ère année - Filière Info
PG104

TD n°1 - Expressions Scheme

Exercice 1: Mise en place de l'environnement

1. Ajouter le chemin `/opt/racket/6.11/bin` dans votre variable d'environnement `PATH`.
2. Exécuter le programme `drracket` en ligne de commandes. Dans la fenêtre du programme, sauvegarder votre fichier.

   Remarque : la première ligne du fichier représente le langage utilisé pour l'évaluer, en l'occurrence le langage Racket, à travers la directive `#lang racket`.
3. Copier le code suivant depuis le fichier de sources (disponible depuis la page des tds) vers la fenêtre représentant votre fichier. L'évaluer avec `F5`.

```
;; Computes the factorial of the integer n
(define (factorial n)
  (if (<= n 1)
    1
    (* n (factorial (sub1 n)))))
```

4. Evaluer dans la REPL (*read-eval-print loop* ou boucle d'interaction) la ligne `(factorial 4)`. Placer cette ligne à l'intérieur de votre fichier.
5. Lancer le débuggueur (bouton *Debug*) et appuyez sur *Step*.

▷ Quelques commandes pour DrRacket utiles :

`C-z`	Annuler la dernière modification (*undo*)
`C-s`	Sauvegarde le fichier en cours
`C-f`	Effectue une recherche dans le fichier en cours
`F5`	Évalue le fichier courant dans la REPL
`C-M-;`	Commenter un bloc de code
`C-M-=`	Décommenter un bloc de code
`C-k`	Forcer le programme à quitter

Remarque : `C` correspond usuellement à la touche "Ctrl" du clavier, et `M` à la touche "Alt" ou "Esc". Dans une commande, les touches séparées par un tiret doivent être tapées simultanément, sinon il faut les taper les unes à la suite des autres.

Exercice 2: Prise en main
Parmi les expressions suivantes, lesquelles sont correctes en Scheme ?

- `- 2 1`
- `(+ (*) (+))`
- `(* (+ 2 5) ((* 3 4) (- 1 2)))`
- `(12 + (* 3 4))`
- `(+ 2 (* 5 (log (8))))`
- `(+ 1 2 3 4 5 6)`
- `(log -2)`
- `(define 3 (+ 2 1))`

Deviner leurs valeurs avant de les entrer dans la boucle d'interaction, les corriger si besoin est, puis les évaluer avec la commande `F5`.

Exercice 3: Quelques fonctions
Définir et *tester* les fonctions suivantes :

- `square` (calcule le carré d'un nombre),
- `mean` (calcule la moyenne de deux nombres),
- `eval-quadratic` (évalue un trinôme $ax^2 + bx + c$ donné par ses coefficients a, b et c en un point x donné).

Exercice 4: Prise en main (suite)
Définir les symboles `a`, `b`, `c`, `d`, `e` ci-dessous en créant alternativement des constantes et des fonctions sans paramètres :

- $a() = \sqrt{1 + \sqrt{2 + \sqrt{3}}}$
- $b = (2 + 3).(4 + 5 + 6)$
- $c() = \log\left(99^2 + 3\right)$
- $d = \frac{a()+b}{a()-b}$
- $e() = \frac{a()+b}{a()+2b} - \sqrt{\frac{a()+2b}{a()+b}}$

Exercice 5: Racines de trinôme - Utilisation de cond/if

1. Écrire une fonction `discriminant` qui, étant donnés 3 arguments a, b et c représentant les coefficients d'un trinôme, calcule le discrimant de ce trinôme
2. Écrire deux fonctions `racine1` et `racine2` qui, étant donnés 3 arguments a, b et c représentant les coefficients d'un trinôme, calculent respectivement la valeur de la première racine et de la deuxième la racine du trinôme. Au cas où il n'y a qu'une racine, les deux fonctions renvoient la même valeur
3. Écrire une fonction `carac-racines-trinome` qui, étant donné 3 arguments a, b et c représentant les coefficients du trinôme, indique si le trinôme aura 2 racines réelles, 1 racine réelle ou 2 racines complexes. Faire une version avec des `if` puis une version avec `cond`.

Exemple d'utilisation :

```
(carac-racines-trinome 1 3 1) ;; → "Two distinct real roots"
(carac-racines-trinome 1 2 1) ;; → "One real root of multiplicity 2"
(carac-racines-trinome 1 1 1) ;; → "Two complex conjugate roots"
```

Pour aller plus loin...

Exercice 6: Bases numériques

1. Convertir en utilisant un algorithme *récursif* un nombre entier en la chaîne de caractères de sa représentation binaire.

   Exemple d'utilisation :

   ```
   (convert 666) ;; → "1010011010"
   ```

   Remarque : les pages de la documentation concernant les nombres (`https://docs.racket-lang.org/reference/numbers.html`) et les chaînes de caractères (`https://docs.racket-lang.org/reference/strings.html`) contiennent des fonctions nécessaires à l'écriture de ces fonctions.

2. Étendre cette fonction pour qu'elle fonctionne en n'importe quelle base ≤ 9. Pour tester votre fonction, écrire la fonction inverse qui prend une chaîne de caractères et une base et calcule la valeur entière correspondante.