

Fiche de TP no. 1

Caveats : *le temps passe vite ! Si, après vingt minutes, vous êtes encore au premier exercice, cela est un bon indice que vous ne vous donnez pas les moyens d'apprendre.*

Objectifs : *apprendre à utiliser ghci, ghc, runghc, avec un éditeur de texte ; écrire des scripts et les modifier ; représenter des simples problèmes mathématiques dans Haskell ; savoir typer les expressions, face à l'ordinateur.*

Exercice 1. Parcourez les transparents du cours (en particulier le ch02) avec l'interprète ghci à la main.

Exercice 2. Écrivez votre premier script HelloWorld.hs et exécutez ce script via ghci, runghc, et en utilisant le compilateur ghc.

Pour ce faire, veuillez aussi choisir un éditeur de texte (ou un IDE) capable de reconnaître et gérer la syntaxe du langage Haskell (voir les pages <https://wiki.haskell.org/IDEs> et <https://wiki.haskell.org/Editors>).

Exercice 3. Le script ci-dessous contient deux erreurs de syntaxe :

```
N = a 'div' length xs
  where
    a = 10
    xs = [1,2,3,4,5]
```

En rappelant ce qui a été dit en cours,

1. trouvez ces erreurs,
2. rappelez quelle est la règle générale qui n'est pas respectée,
3. corrigez-les, de façon à pouvoir compiler le script et le tester avec ghci.

Attention : lorsque on fait un copier-coller, le caractère ‘ (backquote, apostrophe inversée) n'est pas d'habitude bien transcrit et donne lieu à un troisième erreur.

Exercice 4.

1. Écrivez, sur une feuille, les définitions des fonctions factoriel et coefficient binomial.
2. Donnez ces mêmes définitions dans un script Haskell.
3. Corrigez la syntaxe du script et chargez le script dans l'interprète.
4. Testez, avec l'interprète, la correction de votre script avec des valeurs connus de ces fonctions.

Rappel : vous pouvez utiliser les fonctions `product :: Num a => [a] -> a` et `enumFromTo :: Int -> Int -> [Int]`.

Exercice 5 : Listes.

1. Écrivez un script Haskell contenant la définition d'une fonction qui calcule la moyenne d'un ensemble de valeurs réels passés en paramètre sous la forme de liste de flottants.

Attention : la fonction `length` retourne un entier, et l'opérateur `/` demande deux flottants en tant que opérandes. C'est-à-dire, `length :: [a] -> Int` et `(/) :: Fractional a => a -> a -> a`. Utilisez la fonction `fromIntegral :: (Num b, Integral a) => a -> b` pour transtyper un entier vers un flottant.

La page https://wiki.haskell.org/Converting_numbers documente les transtypes numériques de Haskell.

2. L'écart moyen d'une suite de réels x_1, \dots, x_n peut se définir comme la moyenne de la suite $\sqrt{(x_i - \mu)^2}$, $i = 1, \dots, n$, où μ est la moyenne de x_1, \dots, x_n . Ajoutez au script la définition d'une fonction qui calcule l'écart moyen d'une suite.

Pour ce faire, vous pouvez vous servir de la fonction `map :: (a -> b) -> [a] -> [b]`, qui applique une fonction `f` à tous les éléments d'une liste. Par exemple, `map fst [(1,2),(3,4),(5,6)]` s'évalue à `[1,3,5]`.

3. Testez la correction du script avec des valeurs connues.

Exercice 6 : Fonctions. Recopiez¹ les définitions suivantes dans un script `Haskell` :

```
second xs = head (tail xs)
swap (x,y) = (y,x)
pair x y = (x,y)
mult x y = x * y
double = mult 2
palindrome xs = reverse xs == xs
twice (f,x) = f (f x)
```

1. En utilisant la commande `:type`, demandez à l'interprète le type des toutes les fonctions définies.
2. Quelles sont les fonctions de deux arguments ? Si une telle fonction n'est pas en forme curriifiée, donnez une définition équivalente qui soit en forme curriifiée.
3. Quelles sont les fonctions d'ordre supérieur ? Testez-les.
4. Quelles sont les fonctions polymorphes ? Testez ces fonctions avec des objets passés en paramètre de types différents.

Exercice 7 : Tuples. On peut représenter un nombre complexe comme un couple de nombres réels, la partie réelle et la partie imaginaire.

1. En utilisant cette représentation des complexes, écrivez un script où les opérations élémentaires sur les nombres complexes—`somme`, `produit`, `division`²—soient définies.
2. Ajoutez, dans le script avant la définition de chaque fonction, une déclaration du type de la fonction.
3. Une fois chargé le script avec `ghci`, évaluez les expressions suivantes :

$$i(1+i) + 1-i, (1+i)(2+3i), \sum_{n=1}^{30} ni.$$

4. Ajoutez dans le script la définition des fonctions `module` et `angle`.

1. Le code source pour cet exercice se trouve ici : http://pageperso.lif.univ-mrs.fr/~luigi.santocanale/teaching/PF/code/TP1/exo_TP1-Fonctions.hs

2. Utilisez [Wikipedia](#) si vous ne vous rappelez pas les opérations sur les nombres complexes.