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 editeur de texte (ou un IDE) capable de reconnaître et gérer la syntaxe du langage `Haskell` (voir la page 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 facon à pouvoir compiler le script et le tester avec `ghci`.

(Attention : il se peut qu'en faisant de copie-coller, le caractère ` ne soit pas bien transcrit et donne lieu à un troisième erreur.)

Exercice 4.

1. Écrivez, sur une feuille, les définitions du factoriel et du 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.

Exercice 5 : *Listes.* Écrivez un script Haskell contenant les définitions des deux fonctions qui calculent la moyenne et l'écart moyen d'un ensemble de valeurs réels passés en paramètre sous la forme de liste de flottants. Testez le script.

Exercice 6 : *Fonctions.* Recopiez [1] 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.

1. Le code source pour cet exercice se trouve ici : http://pageperso.lif.univ-mrs.fr/~luigi.santocanale/teaching/PF/code/exo_TP1_Fonctions.hs

2. Quelles sont les fonctions de deux arguments ? Si une telle fonction n'est pas en forme currifiée, donnez une définition équivalente qui soit en forme currifiée.
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 : *Tuplets.* On peut représenter un nombre complexe comme un couple de nombres réels, la partie réelle et la partie imaginaire.

1. En utilsant 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`, evaluez 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`.

2. Utilisez `Wikipedia` si vous ne vous rappelez pas les opérations sur le nombres complexes.