

## Fiche de TP no. 1

**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`.

**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`.

**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.** (Tuplets) 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<sup>1</sup>—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`.

**Exercice 7.** (Fonctions) Recopiez 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. Utilisez Wikipedia si vous ne vous rappelez pas les opérations sur les nombres complexes.

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.