Programmation fonctionnelle
TD n° 1 : Programmer avec des fonctions

Exercice 1 (Types et fonctions)

Déterminez, si possible, le type des fonctions suivantes :

1. `let f1 x = if x < 0 then -x else x`
2. `let f2 x = x / 0`
3. `let f3 x = (x + 1, x +. 1.0)`
4. `let f4 x y z = if x then y + 1 else z`
5. `let f5 x y z = if x then y else z`
6. `let f6 x y = x y`
7. `let rec f7 x = f7 x`
8. `let rec f8 x = if true then f8 x else 0`

Exercice 2 (Exponentiation entière)

On rappelle que l'exponentiation `**` n'est définie que sur les flottants.

1. Définissez une fonction `exp1` qui réalise l'exponentiation entière à partir de l'exponentiation flottante et de conversions.
2. Définissez une fonction `exp2` qui réalise l'exponentiation entière à partir du schéma de récurrence :
$$x^0 = 1 \qquad x^{n+1} = x * x^n$$
3. Définissez une fonction `exp3` qui réalise l'exponentiation entière à partir du schéma de récurrence :
$$x^0 = 1 \qquad x^{2n} = (x * x)^n \qquad x^{2n+1} = x * (x * x)^n$$
4. En modifiant la fonction précédente, comment obtiendriez-vous les trois derniers chiffres du nombre 2019^{2019} ?

Exercice 3 (Intégration)

Une des applications courantes de l'informatique concerne l'approximation numérique de différents problèmes. On s'intéresse ici au calcul d'une valeur approchée de

$$\int_a^b f(t)dt$$

où f est une fonction continue sur l'intervalle $[a, b]$.

Une formule de quadrature est une approximation de l'intégrale à l'aide d'une formule linéaire utilisant un nombre fini de valeurs prises par la fonction f associées à des poids.

Formule du rectangle	$f(a) * (b - a)$
Formule du point milieu	$f\left(\frac{a+b}{2}\right) * (b - a)$
Formule du trapèze	$\frac{f(a)+f(b)}{2} * (b - a)$
Formule de Simpson	$\left(\frac{1}{6}f(a) + \frac{2}{3}f\left(\frac{a+b}{2}\right) + \frac{1}{6}f(b)\right) * (b - a)$

1. Quels sont les paramètres nécessaires pour calculer l'intégrale ? En déduire le type Ocaml que doit avoir une formule de quadrature.
2. Implémentez la formule de Simpson.
3. L'erreur produite par les formule de quadrature est croissante avec l'amplitude de l'intervalle d'intégration. Pour limiter l'erreur, on profite de la linéarité de l'intégration :

$$\int_a^b f(t)dt = \sum_{i=0}^{n-1} \int_{a_i}^{a_{i+1}} f(t)dt$$

où $a = a_0 < a_1 < \cdots < a_{n-1} < a_n = b$. On utilise alors une formule de quadrature sur chacun des sous-intervalles obtenus (dont l'amplitude est plus faible). Que vaut le terme a_i si l'on veut découper de manière régulière l'intervalle $[a, b]$?
4. On veut définir une fonction `quadrature_composite` qui réalise l'optimisation précédente. La formule de quadrature utilisée ainsi que le nombre de sous-intervalles sont passés en paramètres. Quelle est le type de la fonction ?
5. Définissez la fonction `quadrature_composite`.

Exercice 4 (Mot de Fibonacci)

En théorie des langages, le mot de Fibonacci est la limite de la suite de chaînes binaires définie par :

$$S_0 = 0 \qquad S_1 = 01 \qquad S_{n+2} = S_{n+1}S_n$$

Les premiers termes de la suite sont donc :

S_0	0
S_1	01
S_2	010
S_3	01001
S_4	01001010
S_5	0100101001001
$\ldots$	$\ldots$

1. Définissez la fonction `mot_fibonacci` qui renvoie S_n. Prenez garde à la complexité de la fonction.
2. Un morphisme de mots est une fonction qui transforme un mot en un autre et qui est compatible avec la concaténation. Autrement dit, une fonction ϕ est un morphisme si pour tous mots u et v, $\phi(uv) = \phi(u)\phi(v)$. Un morphisme est donc entièrement défini par l'image des lettres : pour tout mot $u = u_0 \ldots u_{n-1}$, $\phi(u) = \phi(u_0) \ldots \phi(u_{n-1})$. On peut donc les représenter par le type `char -> string`.
   Définissez une fonction `appliquer_morphisme` qui prend en paramètre un morphisme ϕ et un mot u et qui renvoie $\phi(u)$.
3. Définissez une fonction `appliquer_morphisme_itere` qui prend en paramètre un morphisme ϕ, un mot u et un entier n et qui renvoie $\phi^n(u)$.
4. La suite S peut être définie à l'aide d'un morphisme :

$$S_0 = 0 \qquad S_{n+1} = \phi(S_n)$$

où ϕ est le morphisme tel que $\phi(0) = 01$ et $\phi(1) = 0$.
Déduisez-en une nouvelle implémentation de la fonction `mot_fibonacci`.