Programmation fonctionnelle
TP n° 1 : Programmer avec des fonctions

Exercice 0 (Familiarisation avec l'environnement)

Vérifiez que votre environnement est correctement configuré. Dans un terminal tapez la commande `ocaml -version`. La réponse devrait être `The OCaml toplevel, version 4.07.0`

Lancez maintenant le logiciel `emacs`, ce sera notre éditeur privilégié. Vérifiez que les menus Tuareg et Merlin apparaissent dans la barre des menus.

Emacs est connu pour avoir des raccourcis inhabituels (particulièrement pour un utilisateur venant de Windows). Vous trouverez ci-après ceux que nous utiliserons le plus. Les lettres `C` et `M` correspondent respectivement aux touches `Ctrl` et `Alt` du clavier. Des combinaisons de touches successives peuvent apparaître, par exemple `C-x C-s` indique qu'il faut appuyer sur les touche `Ctrl` et `x` simultanément puis sur les touches `Ctrl` et `s` simultanément.

Raccourci	**Effet**
`C-x C-f`	Ouvrir un nouveau fichier
`C-x C-s`	Sauvegarder
`C-x C-w`	Sauvegarder sous
`C-x u`	Annuler
`C-w`	Couper
`M-w`	Copier
`C-y`	Coller
`C-x h`	Tout sélectionner
`C-c C-s`	Lancer l'interpréteur Ocaml
`C-c tab`	Interrompre l'évaluation courante de l'interpréteur
`C-c C-e`	Interpréter l'expression courante
`C-c C-b`	Interpréter le fichier courant
`C-c C-t`	Montrer le type de l'expression courante

De plus la touche Tab permet de réindenter automatiquement la ligne ou la sélection courante.

Vous pouvez maintenant écrire quelques expressions au toplevel pour en voir le résultat. Faites de même depuis emacs en demandant à interpréter le contenu de votre fichier. N'hésitez pas à appelez votre enseignant si vous rencontrez un problème.

Exercice 1 (Fonctions usuelles)

Définissez les fonctions suivantes :

1. La fonction `val_abs` qui renvoie la valeur absolue d'un réel.

2. La fonction `signe` telle que `signe x` renvoie `0` si `x = 0`, `1` si `x > 0` et `-1` sinon.
3. La fonction `fac` qui calcule la factorielle de son argument.
4. La fonction `est_diviseur` telle que `est_diviseur n d` renvoie **`true`** si `d` divise `n` et **`false`** sinon.
5. La fonction `est_premier` qui indique si son argument est un nombre premier. Pensez à définir une fonction auxiliaire interne.

Exercice 2 (Hello world !)

1. Écrivez une expression qui affiche `Hello world !`. Quelle est la valeur de cette expression ?
2. Définissez une fonction `hello` qui affiche `Hello world !`.
3. Modifiez la fonction `hello` pour qu'elle prenne en paramètres un nom et un âge et qu'elle affiche une petite phrase de présentation. Par exemple, `hello2 "Julien" 34` affiche

   `Hello, my name is Julien and I am 34.`

   Indication : La fonction `string_of_int` pourrait vous être utile.

Exercice 3 (Approximation de Pi)

On représente un point du plan par un couple de nombres flottants (x, y).

1. Définissez une fonction `distance` qui calcule la distance d'un point du plan à l'origine du repère.
2. On considère le cercle de rayon 1 centré à l'origine du repère et le carré circonscrit dont les côtés sont parallèles aux axes. Les sommets ont donc pour coordonnées $(1, 1)$, $(1, -1)$, $(-1, -1)$ et $(-1, 1)$. Quel est la probabilité qu'un point tiré aléatoirement à l'intérieur du carré soit aussi à l'intérieur du cercle ?
3. Définissez une fonction `random_point` qui renvoie aléatoirement un point à l'intérieur du carré.
4. Définissez une fonction `approche_pi n` qui tire au hasard `n` point à l'intérieur du carré, compte combien sont dans le cercle et en déduit une approximation de π.

Exercice 4 (Zéro d'une fonction)

On considère une fonction f continue sur un intervalle $[a, b]$ telle que $f(a)f(b) < 0$. On sait alors qu'il existe un réel c tel que $f(c) = 0$. On se propose de trouver un tel réel à une précision donnée près. Plus précisément, on va programmer une fonction `approche0 f a b p` qui renvoie un nombre c tel que $|f(c)| < p$.

On procède à une recherche dichotomique. On pose $c = (a + b)/2$. Si $|f(c)| < p$, on renvoie c. Si $f(a)f(c) < 0$, on poursuit la recherche dans $[a, c]$ sinon on poursuit la recherche dans $[c, b]$.

1. Définissez la fonction `approche0`.
2. Que calcule le code suivant ?

```
let f x = x *. x -. 2.0 in
approche0 f 1.0 2.0 0.000001
```

3. Déduisez en une nouvelle façon d'approximer le nombre π.

Exercice 5 (Préfixes, suffixes, facteurs)

On dit qu'un mot u est préfixe d'un mot v si v commence par le mot u. C'est un suffixe si v se termine par le mot u et c'est un facteur si u apparaît à l'intérieur du mot v à une position quelconque.

1. Définissez une fonction `est_facteur_position u v i` qui renvoie **`true`** si `u` est un facteur de `v` à partir de la position `i` et **`false`** sinon.
2. Déduisez-en les fonctions `est_prefixe`, `est_suffixe` et `est_facteur`.

Exercice 6 (Curryfication)

1. Rappelez ce qu'est la curryfication.
2. On considère des fonctions d'arité 2 dont les arguments et le résultat sont entiers.
   (a) Donnez le type d'une telle fonction si elle est currifiée.
   (b) Donnez le type d'une telle fonction si elle n'est pas currifiée.
   (c) Définissez une fonction `curry` qui prend en paramètre une fonction non currifiée et renvoie sa version currifiée. Définissez de même la fonction `uncurry` qui en est la réciproque.
   (d) Les types donnés par l'interpréteur Ocaml pour les fonctions `curry` et `uncurry` sont-ils ceux attendus ?