Remarques communes à tous les TPs

Jean-Baptiste Rouquier

1 Avant le premier TP

1.1 Aspects pratiques

Tous les sujets de TP seront faits en OCaml. Si vous êtes déjà habitués à camllight, manquez cruellement de temps, êtes prêts à traduire en camllight tout ce qui se passe en TP et envisagez d'arrêter l'info après les concours, vous pouvez conserver camllight. Sinon je vous recommande fortement OCaml, plus puissant (nombre de bibliothèques disponibles incomparable, compilation native, *profiling*...), plus répandu, mis à jour... Le choix entre camllight et OCaml n'influera pas sur l'aide aportée ni la notation.

Le manuel officiel de Caml se trouve sur la page caml.inria.fr/pub/docs/manual-ocaml , que vous pouvez ouvrir depuis le navigateur internet Mozilla. La section la plus utile sera «The standard libray» dans la partie IV «The Objective Caml library». Je vous invite en particulier à consulter les fonctions des modules `Array`, `List`, `Printf`, `Queue`, `Random` et `Stack` pour éviter de réinventer la roue.

1.2 Partie spécifique aux séances aux lycée Janson de Sailly

Votre login est la première lettre de votre prénom suivie de votre nom, sans espaces. Votre mot de passe est «123». Choisir KDE si vous ne connaissez pas les autres gestionnaires de fenêtres et annulez la configuration du gestionnaire de mail.

Changez votre mot de passe lors de votre premier login : choisissez un mot de passe (que personne ne peut deviner, en particulier pas de prénom ou nom de rue, pas un mot du dictionnaire, au moins six lettres ou chiffres, vous pouvez prendre les initiales d'une phrase) ; puis ouvrez une console (le petit rectangle noir en bas à gauche, représentant un écran) et tapez

```
yppasswd
```

Recopiez ensuite les bons fichiers de configuration : tapez

```
cd
cp ~jrouquie/.{emacs,zshrc,bashrc,bash_profile} .
```

(sans oublier le dernier point précédé d'un espace), fermez la console et rouvrez-en une nouvelle. Emacs devrait maintenant connaître le tuareg-mode. Pour le vérifier, ouvrez un fichier temp.ml :

```
emacs temp.ml &
```

tapez une ligne de Caml (`let a = 5` par exemple) et appuyez sur F12 pour compiler[1]. Emacs vous proposera peut-être camllight, tapez ocaml à la place.

Si vous vous trompez de commande dans Emacs, tapez Ctrl+g pour annuler la commande en cours.

Votre code source doit s'appeler login-tpx.ml où «x» est le numéro du TP et «login» est votre login. Si vous souhaitez lui donner un autre nom, c'est en tout cas le nom qu'il doit avoir quand vous me l'enverrez (voir paragraphe suivant). Si vous travaillez à deux sur la même machine, mettez en commentaire le nom ou login des deux personnes en tête du code source.

Pour lire le sujet à l'écran, tapez

```
xpdf ~jrouquie/sujets/foo.pdf &
```

[1] la commande définie par tuareg est Ctrl+c Ctrl+e, mais sur ces ordinateurs le raccourcis F12 a été ajouté.

où foo est le nom du sujet du jour. Le «`&`» permet de continuer à taper des commandes dans la console.

N'hésitez pas à poser des questions pendant le TP : sur le sujet, la syntaxe, les fonctions de Caml, le TP précédent, une fonction qui ne compile pas... Je suis là pour ça, et cela ne peut que montrer votre motivation (et ne peut pas faire baisser votre note).

Cinq minutes avant la fin du TP, recopiez votre code source dans `~jrouquie/remises/login-tpx.ml` en remplaçant votre login et le numéro du TP :

```
cp login-tpx.ml ~jrouquie/remises/
```

Important : *déloguez vous avant de partir* (clic sur l'icône en bas à gauche et «Terminer la session»).

1.3 Consignes de rédaction

Les fonctions doivent être commentées chaque fois que nécessaire. Il est également demandé de les tester, c'est l'un des meilleurs moyens de s'autocorriger. Vérifiez que le type est compatible avec celui demandé.

Il vaut mieux implémenter une fonction avec une mauvaise complexité que ne pas l'implémenter du tout.

2 Erreurs fréquentes

2.1 sur les relations

Ne pas confondre

- ***commutatif*** qui qualifie une opération de composition interne, comme l'addition dans les espaces vectoriels. Par exemple en maths l'addition des naturels est une opération $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$, elle est commutative :
$$\forall x, y \quad x + y = y + x$$
En Caml une opération sur les naturels peut avoir pour type `int -> int -> int`.
- ***symétrique*** qui caractérise une relation $\mathcal{R}$ telle que
$$\forall x, y \quad x\mathcal{R}y \iff y\mathcal{R}x$$
En Caml une relation sur les naturels (comme l'égalité) peut avoir pour type `int -> int -> bool`.
- ***réflexif*** qui caractérise une relation $\mathcal{R}$ telle que
$$\forall x \quad x\mathcal{R}x$$

Une relation est non réflexive si et seulement si $\exists x \quad \neg x\mathcal{R}x$. Un graphe vérifie la propriété plus forte $\forall v \in V \quad \neg(x\mathcal{R}x)$. ($\mathcal{R}$ est la relation d'***adjacence*** : $x\mathcal{R}y$ signifie «il y a un arc de x à y».)

2.2 sur la syntaxe

Le premier élément d'un tableau a l'indice zéro.

Un piège subtil :

```
let empty = [];;
let teste liste = match liste with
 | empty -> "vide"
 | t :: q -> "non vide"
```

renvoie toujours `"vide"`. En effet le second cas du filtrage crée une variable `t` contenant la tête de la liste et une variable `q` qui en contient la queue. Le premier cas crée de même une (nouvelle) variable `empty` dont la valeur est celle de `liste`, au lieu de tester l'égalité à l'ancienne variable `empty`. D'ailleurs le compilateur prévient «Warning : this match case is unused.» car le second cas ne sera jamais utilisé. Solution :

```
let empty = [];;
let teste liste = match liste with
 | lst when lst = empty -> "vide"
 | t :: q -> "non vide"
```

« `:=` » n'est utilisé que pour modifier une référence. « `<-` » sert à modifier une case d'un tableau ou à modifier un champ mutable d'un enregistrement.

2.3 soigner son style

Éviter d'utiliser 0 et 1 au lieu de **true** et **false** : un `int array` qui ne contient que des 0 et des 1 devrait souvent être plutôt un `bool array`.

Dans un tableau de type `int option array`, on ne peut stocker que des `int option`, jamais des `int` seuls (au lieu de `42` il faut écrire `Some 42`). Le type `option` permet de distinguer entre une valeur valide et une valeur manquante (non encore calculée, erreur, sans signification à cet endroit, etc.). Cela évite d'utiliser des valeurs du même type en leur attribuant une signification particulière. Par exemple, renvoyer -1 quand un entier positif est attendu est souvent un moyen de signaler une erreur ou une absence de réponse. Mais cette dernière façon de procéder est source de bugs, on préferera donc le type `option`.