TP3 Caml : D'autres outils...

Sujet à finir pour le dimanche 05/10/2014. À rendre par mail à
aurelie.lagoutte@ens-lyon.fr

1 Arbres binaires de recherche

Un *arbre binaire de recherche* est un arbre dont les noeuds sont étiquetés par des entiers, et tel que pour chaque noeud x, les étiquettes apparaissant dans le sous-arbre gauche de x sont inférieures à l'étiquette de x, et les étiquettes apparaissant dans le sous-arbre droit de x sont supérieures à l'étiquette de x. Il est dit *équilibré* si pour chaque noeud, les sous-arbres droit et gauche ont la même taille (plus ou moins 1).

Exercice 1 (Arbres binaires) *On reprend le type* `arbre` *défini au TP1.*

1. *Écrivez une fonction renvoyant la liste des nœuds d'un arbre binaire, lus en ordre infixe (gauche-nœud-droite). On pourra utiliser la fonction* `List.append`*.*
2. *À l'aide de la fonction tri-fusion du TP1, écrivez une fonction prenant un arbre binaire d'entiers et renvoyant un arbre binaire équilibré trié contenant les mêmes entiers : on pourra décomposer en trois étapes : obtenir la liste des étiquettes, la trier, puis construire un arbre binaire de recherche équilibré à partir de cette liste.*
3. *Écrivez une fonction de recherche d'un élément dans un arbre trié, donnant la profondeur à laquelle se trouve l'élément le cas échéant (-1 s'il n'y est pas).*

2 Tableaux et références

L'exemple suivant illustre l'utilisation de tableaux en `oCaml` (on passe ici dans le monde impératif, par opposition au monde récursif).

```
let digits = Array.create 10 0;;
for i = 0 to 9 do digits.(i) <- digits.(i) + i done;;
let chriffre_trois = digits.(3);;
for i = 0 to 9 do print_int(digits.(i)) done; print_newline();;
```

En `Caml`, la valeur d'une variable n'est pas modifiable. On peut utiliser des références pour émuler les pointeurs du langage `C`.

```
let x = ref 0;;
x := !x +1;;
x;;
```

Exercice 2 (Swap) *Écrivez une fonction* `swap : 'a ref -> 'a ref -> unit` *telle que* `swap x y` *échange les contenus des références x et y (en faire une version avec variable auxiliaire et une sans, lorsque x et y contiennent des entiers).*

Exercice 3 (Nombre mystère) *Écrivez une fonction* `mystere` *de type* `unit->unit` *implémentant le jeu du nombre mystère : un nombre est choisi aléatoirement*[1] *entre* 0 *et* 100 *et l'utilisateur dispose de* 7 *essais pour deviner ce nombre. On indiquera à chacun de ses essais si le nombre proposé est plus grand, plus petit ou égal au nombre mystère.*

Exercice 4 *Chance Écrivez une fonction* `chance` *de type* `unit->unit` *qui remplit un tableau de taille 10 avec des entiers aléatoires de 0 à 10. L'utilisateur gagne (i.e. reçoit un message de victoire) lorsqu'il existe un ou plusieurs indices* `i` *tels que t.(i)=i. Sinon, il reçoit un message de défaite.*

3 Enregistrement

Caml possède une structure nommée *enregistrement* qui se rapproche légèrement de la programmation orientée objet. Testez le code suivant :

```
type client={nom:string; adresse: string; mutable solde:int; mutable anciennete:int};;

let tom={nom="Tom"; adresse= "Ici"; solde=0; anciennete=1};;

tom.nom;;

tom.solde<-4;;
tom.solde<-tom.solde+1;;
tom.adresse<-"Paris";;
```

Qu'en déduisez-vous sur le mot clé `mutable` ?

Exercice 5
- *Construisez un type* `etudiant` *contenant un nom, un prénom, et une liste de notes.*
- *Écrivez une fonction* `ajoute: etudiant -> int-> unit` *qui ajoute une note à un étudiant.*
- *Écrivez une fonction* `moyenne: etudiant -> float` *qui calcule la moyenne d'un étudiant.*
- *Écrivez une fonction* `triche: etudiant -> unit` *qui enlève la plus mauvaise note d'un étudiant.*
- *Écrivez une fonction* `moyenne_classe: etudiant list -> float` *qui calcule la moyenne de classe (prendre la première note de chaque liste).*

4 Graphisme en `Caml`

La bibliothèque `graphics` permet de dessiner des images, de détecter des mouvements de souris et des frappes de touches. Elle permet même de produire des sons !
Une description des fonctions disponibles se trouve à l'URL suivante :

`http://caml.inria.fr/pub/docs/manual-caml-light/node16.html`

Éssayez d'analyser et tester ce que fait le code suivant :

1. On utilisera la fonction `Random.int n` que l'on initialisera d'abord avec `Random.self_init()`.

```
#load "graphics.cma";;
Graphics.open_graph "";;
let rec draw l = match l with
   [] -> ()
  |(x0,y0)::l1 -> Graphics.lineto x0 y0;
                  Graphics.moveto x0 y0;
                  draw l1;;
```

Exercice 6 (Damier)
Écrivez une fonction prenant en entrée deux entiers m et n et qui dessine un échiquier de taille m par n. On utilisera `fill_rect`*, et on construira des carrés de taille 100 par 100. Bonus : affiner ceci en adaptant la taille des carrés (on utilisera les fonctions* `size_x, size_y`*).*

Exercice 7 (Flocon de Koch) *Le flocon de Koch est un exemple très classique de fractale. On l'obtient de la manière suivante :*

- *Partir d'un triangle équilatéral (plutôt grand) ;*
- *Pour chaque côté :*
  - *Le subdiviser en trois segments de longueurs égales ;*
  - *Poser un nouveau triangle équilatéral sur le segment du milieu, à l'extérieur du grand triangle ;*
  - *Effacer ce "segment du milieu".*
- *Répéter récursivement sur les 12 segments restants.*

Écrivez une fonction qui effectue un nombre d'itérations donné de la construction du flocon de Koch, et testez cette fonction sur un nombre significatif d'itérations.