Travaux Pratiques ML No.5

Cours d'Informatique de Deuxième Année

—Deug SSM M4—

Aujourd'hui...

L'organisation de ce TP est la suivante:

— Les arbres binaires sur les entiers.
— Les arbres de type polymorphe.
— Fonctionnelles d'itération sur les arbres.
— Quelques algorithmes...

Arbres binaires sur les entiers

On donne le type des arbres binaires sur les entiers.

```
# type bintree = Niltree | Leaf of int | Node of int * bintree * bintree ;;
```

▸ **Exercice 2**. Donnez la fonction qui calcule la hauteur d'un arbre binaire.

. . . . . .

▸ **Exercice 3**. Donnez la fonction qui calcule la somme de tous entiers contenus dans l'arbre.

. . . . . .

▸ **Exercice 4**. Ecrire les fonctions de parcours préfixe, infixe et suffixe d'un arbre binaire.

. . . . . .

. . . . . .

. . . . . .

▸ **Exercice 5**. Ecrire la fonction d'insertion d'un entier dans un arbre binaire de recherche.

. . . . . .

. . . . . .

▸ **Exercice 6**. Donnez la fonction de suppression d'un élément dans un arbre binaire de recherche.

. . . . . .

. . . . . .

. . . . . .

▸ **Exercice 7**. Définir une fonctionnelle d'itération sur les arbres binaires de recherche et l'utiliser pour écrire la fonction qui teste si un élément appartient bien à un arbre binaire.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

On définit les arbres binaires sur les expressions arithmétiques de la manière suivante:

```
# type ABExpr = Leaf of int | Node of Op * ABExpr * ABExpr
# and Op = Plus | Moins | Div | Mult ;;
```

▸ **Exercice 8**. Définir une fonctionnelle d'itération `ABExpr_it` sur le type `ABExpr`.

. . . . . .

. . . . . .

. . . . . .

▸ **Exercice 9**. Ecrire la fonction d'évaluation d'une expression de type `ABExpr` d'abord sans `ABExpr_it` et ensuite avec la fonctionelle d'itération.

. . . . . .

. . . . . .

▸ **Exercice 10**. Ecrire une fonction qui traduit un arbre de type `ABExpr` en une `string list` correspondant au parcours préfixe de cet arbre (on dispose de la fonction `string_of_int` en *Caml-Light* qui transforme un entier en une chaîne de caractères).

. . . . . .

. . . . . .

▸ **Exercice 11**. Ecrire une fonction qui transforme une expression de type `string list` en un arbre de type `ABExpr`, la liste constituant un parcours préfixe de l'arbre (on pourra s'aider de la fonction `list_it`; on dispose aussi en *Caml-Light* de la fonction `int_of_string` qui transforme une chaîne de caractères en un entier).

. . . . . .

. . . . . .

. . . . . .

Exercice supplémentaire

▸ **Exercice 12**. Ecrire une fonction de parcours en largeur d'un arbre binaire (on parle aussi de parcours par niveau).