Examen de programmation avancée

ENSIIE, semestre 2

mercredi 30 mars 2011

Durée : 1h45.
Tout document personnel autorisé (pas de prêt entre voisins).
Ce sujet comporte cinq exercices indépendants, qui peuvent être traités dans l'ordre voulu. Il contient 2 pages.
Le barème est donné à titre indicatif, il est susceptible d'être modifié. Le total est sur 20 points.

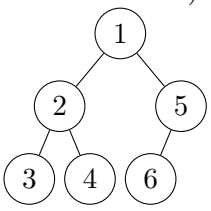
Exercice 1 : Arbres (4 points)

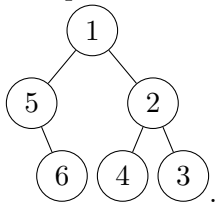
On considère des arbres binaires non-stricts codés en OCaml par le type

```
type arbre = Vide | Noeud of arbre * int * arbre
```

et en C par

```
typedef struct arbre_struct {
  int valeur;
  struct arbre_struct* fg;
  struct arbre_struct* fd;
} * arbre;
```

1. Écrire (au choix en OCaml ou en C) une fonction qui renvoie l'image miroir d'un arbre. Par exemple, 

devient 

. En C, on pourra coder une version destructive.
2. Donner sa complexité en fonction du nombre n de nœuds dans l'arbre.
3. Montrer que le parcours préfixe d'un arbre visite les nœuds dans l'ordre inverse du parcours postfixe de son image miroir.

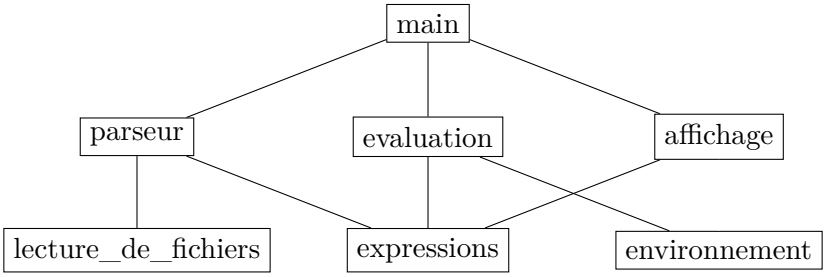
Exercice 2 : Modularité (3 points)

Justifier en quoi la modularité permet d'améliorer :

1. la séparation des tâches ;
2. la réutilisabilité ;
3. la maintenabilité.

Exercice 3 : Makefile (3 points)

Pour écrire un interpréteur d'expressions arithmétiques, on a choisi le découpage en modules suivant :



Un module m1 est relié par une arête vers le bas à un autre module m2 si m1 dépend de (l'interface de) m2. On suppose que chaque module a une interface explicite.

Écrire le `Makefile` correspondant. On pourra considérer au choix que les modules sont des fichiers C ou OCaml. On n'oubliera d'écrire une cible pour produire l'exécutable final.

Exercice 4 : Arbres binaires de recherche (6 points)

1. Décrire un algorithme qui permet de renvoyer la valeur associée à la plus petite clef dans un arbre binaire de recherche non vide.
2. Écrire la fonction correspondante (en OCaml ou en C).
3. Démontrer la correction de cet algorithme (c'est-à-dire qu'il renvoie bien la valeur associée à la plus petite clef quand l'arbre n'est pas vide).
4. Donner en fonction du nombre n de nœuds dans l'arbre sa complexité en moyenne et dans le pire des cas, en considérant des arbres binaires de recherche simples.
5. Même question si on considère des arbres AVL.

Exercice 5 : Choix de structures de données (4 points)

1. On veut utiliser un dictionnaire pour une application web. Le temps de recherche en moyenne doit être le plus rapide possible, mais le temps dans le pire des cas n'est pas important, l'utilisateur changeant sa requête quand le résultat se fait trop attendre. Laquelle des structures vues en cours faut il utiliser ?
2. On veut utiliser un dictionnaire dans une application embarquée synchrone. On doit pouvoir connaître dans tous les cas le temps maximal que prendront les opérations de recherche, d'insertion et de suppression, mais on veut le minimiser. Laquelle des structures vues en cours faut il utiliser ?
3. Décrire une structure de données qui a une complexité en moyenne de $O(1)$ et dans le pire des cas de $O(\log n)$ pour les trois opérations d'insertion, de suppression et de recherche à la fois.