Université Pierre et Marie Curie — Année 2004–2005
MIME — **Examen – 2 heures**
LI101 — 1er Septembre 2005

Aucun document ni machine électronique n'est permis à l'exception de la carte de référence de Scheme. Les téléphones doivent être éteints et rangés dans les sacs.

L'examen dure deux heures. Ce sujet comporte 7 pages.

Les questions peuvent être résolues de façon indépendante. Il est possible, voire même utile, pour répondre à une question, d'utiliser les fonctions qui sont l'objet des questions précédentes.

Répondre sur la feuille même, dans les cadres appropriés. La taille des cadres suggère le nombre de lignes de la réponse attendue (utiliser le dos de la feuille précédente si la réponse déborde des cadres). Le barème (total sur 40) apparaissant dans chaque cadre n'est donné qu'à titre indicatif.

La clarté des réponses et la présentation des programmes seront appréciées. Sauf mention contraire, les fonctions qui apparaîtront dans vos réponses devront être accompagnées de leur spécification.

Ne pas désagrafer les feuilles.

# Exercice 1

Ce problème traite des listes de nombres.

**Question 1.1 –** Écrire une définition de la fonction `ajout1Liste`, qui prend une liste de nombres et renvoie la liste obtenue en ajoutant 1 à tous les éléments de la liste. Ainsi

```
(ajout1Liste '(23 12 5 7 2)) -> (24 13 6 8 3)
```

[3/40]

**Question 1.2 –** Écrire une définition récursive de la fonction `ajout1SiImpairListeRec`, qui prend une liste de nombres entiers et renvoie la liste obtenue en ajoutant 1 à tous les nombres impairs de la liste (et en laissant les autres inchangés). Ainsi

```
(ajout1SiImpairListeRec '(23 12 5 7 2)) -> (24 12 6 8 2)
```

[4/40]

**Question 1.3 –** Écrire une définition de la fonction `ajout1SiImpairListeMap`, qui répond à la même spécification que la fonction `ajout1SiImpairListeRec`, mais dont la définition fait intervenir une fonctionnelle `map`. On définira pour cela une fonction `ajout1SiImpair` qui, étant donné un entier $x$ renvoie $x+1$ si l'entier est impair, et sinon renvoie $x$. Et pour répondre à la question posée, on utilisera cette fonction comme argument d'un `map`.

[3/40]

Section | Numéro d'anonymat

**Question 1.4 –** Écrire une définition de la fonction `ajout1Si` qui prend une liste de nombres $L$ et un prédicat $p$?, et renvoie une liste de même longueur que $L$, où tous les nombres satisfaisant $p$? sont incrémentés de 1. Ainsi

```
(ajout1Si  '(23  12  5  7  2)  even?)  ->  (23  13  5  7  3)
```

[3/40]

**Question 1.5 –** Écrire une définition de la fonction `ajout1ListeKeme` qui, étant donnés une liste de nombres et un entier $k > 0$, renvoie la liste obtenue en ajoutant 1 au $k^{ieme}$ élément de la liste initiale. Et la fonction renvoie une erreur si $k$ est supérieur à la longueur de la liste. Par exemple

```
(ajout1ListeKeme    '(23  12  5  7  2)  3)  ->  (23  12  6  7  2)
(ajout1ListeKeme    '(23  12  5  7  2)  6)  ->  erreur
```

[4/40]

Section | Numéro d'anonymat

**Question 1.6 –** On définit ci-dessous une fonction `mystere`, avec deux fonctions internes `f` et `g`.
Quel est le résultat de l'application de la fonction `mystere` sur la liste `(23 12 5 7 2)` ?
Écrire la spécification de la fonction `mystere` et des fonctions internes `f` et `g`.

```
(define (mystere L)
  (define (f L)
    (if (pair? L)
        (cons (car L)
              (g (cdr L)))
        L))
   (define (g L)
    (if (pair? L)
        (cons (+ 1 (car L))
              (f (cdr L)))
        L))
  ; corps de la fonction mystere
  (f L))
```

[3/40]

**Question 1.7 –** Écrire une définition de la fonction `supprimeListeUnSurDeux`, qui prend une liste et renvoie la liste obtenue en supprimant un élément sur deux dans la liste initiale (en commençant par supprimer le premier). Par exemple

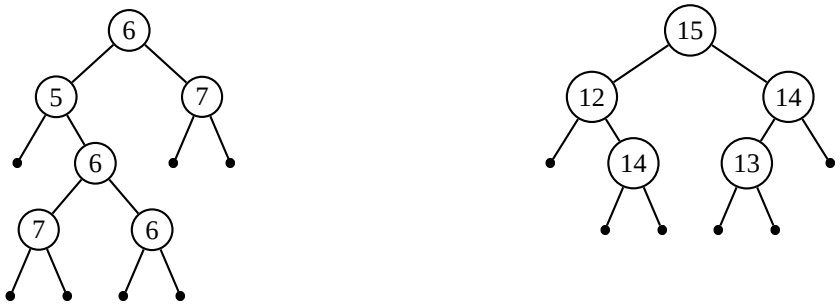
```
(supprimeListeUnSurDeux '(23 12 5 7 2)) -> (12 7)
```

[4/40]

Section | Numéro d'anonymat

## Exercice 2

Ce problème s'intéresse à des arbres binaires étiquetés par des entiers. Les arbres ci-dessous serviront d'exemple par la suite. On suppose que l'arbre de gauche est produit par l'expression `(B1)` et l'arbre de droite par l'expression `(B2)` .




**Question 2.1 –** Écrire une définition de la fonction `somme-etiquettes` , qui prend un arbre binaire étiqueté par des entiers et renvoie la somme de ses étiquettes. Ainsi

```
(somme-etiquettes (B1)) -> 37
(somme-etiquettes (B2)) -> 68
```

[4/40]

Section Numéro d'anonymat

**Question 2.2 –** Un arbre binaire est dit *bien étiqueté* si ses sommets sont étiquetés par des entiers non négatifs tels que pour tout sommet *s* autre que la racine, la valeur absolue de la différence entre l'étiquette de *s* et celle de son père est inférieure ou égale à 1. Écrire une définition du prédicat `bien-etiquete ?`, qui prend un arbre binaire étiqueté par des entiers et renvoie vrai si et seulement si cet arbre est bien étiqueté. Ainsi

```
(bien-etiquete?  (B1))  -> #t
(bien-etiquete?  (B2))  -> #f
```

[5/40]

**Question 2.3 –** On se propose par la suite de construire des arbres binaires bien étiquetés. On va tout d'abord définir la fonction qui permet de calculer une étiquette égale à 1 près à une étiquette donnée.
Écrire une définition de la fonction `proche` , qui étant donné un entier $p$ renvoie un entier $q$ aléatoirement égal à $p-1$ ou $p$ ou $p+1$. Par exemple `(proche 5)` renvoie aléatoirement 4 ou 5 ou 6.
On pourra utiliser la fonction `random`

```
;;; random : int/>=0/ -> int
;;; (random n) renvoie un entier aléatoire entre 0 et n-1
```

[3/40]

**Question 2.4 –** Il s'agit maintenant de construire des arbres binaires bien étiquetés et de forme aléatoire. Pour construire un arbre binaire de taille $n$ (c'est-à-dire contenant $n$ nœuds), on choisit un entier aléatoire $k$ compris entre

Section | Numéro d'anonymat

0 et $n-1$, et l'on construit récursivement deux arbres binaires, respectivement de taille $k$ et $n-1-k$, qui sont les sous-arbres gauche et droit de la racine. Si l'on veut aussi que l'arbre soit bien étiqueté, il faut que, aussi bien pour le sous-arbre gauche que pour le sous-arbre droit, l'étiquette diffère au plus de 1 par rapport à l'étiquette de la racine.

En suivant cette méthode, écrire une définition de la fonction `arbre-etiq-alea` qui, étant donnés un entier positif $n$ et un entier $p$ renvoie un arbre binaire bien etiqueté de taille $n$, dont l'étiquette à la racine est $p$.

[4/40]