

## Devoir sur table – Novembre 2004

LI101

Durée : 1h30

Aucun document ni machine électronique n'est permis à l'exception de la carte de référence de Scheme.

Répondre sur la feuille même, dans les boîtes appropriées. La taille des boîtes suggère le nombre de lignes de la réponse attendue. Le barème (total sur 20) apparaissant dans chaque boîte n'est donné qu'à titre indicatif.

La clarté des réponses et la présentation des programmes seront appréciées. Ne pas désagrafer les feuilles.

### Exercice 1

**Question 1.1** – Écrire, en Scheme, une signature et une définition de la fonction  $H$  ainsi définie sur les entiers naturels :

$$H(n, p) = \begin{cases} H(p, n) & \text{si } n < p \\ n & \text{si } n = p \\ H(n - p, p) & \text{si } n > p \end{cases}$$

[2/20]

**Question 1.2** – Quel est le type de la fonction suivante :

```
(define (mystere x)
  (if (pair? x)
      (or (car x) (mystere (cdr x)))
      (= 1 1) ) )
```

[2/20]

### Exercice 2

Cet exercice tourne autour du triangle de Pascal dont voici les premières lignes.

		1									<i>ligne 0</i>
			1	1							<i>ligne 1</i>
			1	2	1						<i>ligne 2</i>
			1	3	3	1					<i>ligne 3</i>
			1	4	6	4	1				<i>ligne 4</i>
			1	5	10	10	5	1			<i>ligne 5</i>
					...						

Le triangle de Pascal est simplement engendré si l'on remarque que chaque nombre de chaque ligne est la somme des deux nombres situés juste au-dessus et en considérant que tous les nombres hors triangle sont nuls.

**Question 2.1** – Écrire une fonction, nommée `somme-2-a-2`, prenant une liste d'au moins deux nombres quelconques et calculant la liste de leur somme deux à deux. Ainsi

```
(somme-2-a-2 (list 1 2 3)) → (3 5)
```

```
(somme-2-a-2 (list 1 2 3 4)) → (3 5 7)
```

[4/20]

**Question 2.2** – Définir une fonction, nommée `bordee-de-1`, prenant une liste de nombres et encadrant cette liste d'un 1 initial et d'un 1 final. Ainsi

```
(bordee-de-1 (list 2 3 4)) → (1 2 3 4 1)
```

La définition de `bordee-de-1` ne doit comporter qu'un appel au plus à `cons`, qu'un appel au plus à `append` et qu'un appel au plus à `list`.

[2/20]

**Question 2.3** – L'expression `(bordee-de-1 (list))` correspond à la deuxième ligne du triangle de Pascal. Si `L` est une liste contenant la  $n$ -ème ligne du triangle de Pascal, quelle expression permet d'obtenir la  $n + 1$ -ème ligne avec l'aide des fonctions précédentes ?

Section

Groupe

Nom

Prénom

[2/20]

**Question 2.4** – Définir la fonction ainsi spécifiée :

```
;;; pascal-ligne: nat /> 0 / -> LISTE[nat]  
;;; (pascal-ligne i) rend la ligne « i » du triangle de Pascal.
```

Ainsi,

```
(pascal-ligne 2) → (1 2 1)  
(pascal-ligne 3) → (1 3 3 1)
```

[4/20]

Section

Groupe

Nom

Prénom

**Question 2.5** – Définir la fonction, nommée `triangle-pascal`, prenant un entier naturel  $n$  et calculant la liste des premières lignes du triangle de Pascal jusqu'à la ligne «  $n$  ». Ainsi,  
`(triangle-pascal 3)` → `((1) (1 1) (1 2 1) (1 3 3 1))`

[4/20]