Université Bordeaux 1, Sciences et technologies
UFR Informatique et Mathématiques

Maîtrise d'informatique
Session de septembre 2001

Techniques et Fondement de la Programmation
(partie "techniques")

Durée : trois heures (les deux parties)

Tous documents autorisés sauf photocopies illégales de livres. Votre voisin n'est pas un document.

Rappel : Soyez bref et clair ! Personne n'est impressionné par la quantité !

Bon courage !

Exercice 1. (10p)

Voici une fonction Common Lisp :

```
(defun complicated ()
  (let ((result nil))
    (do () (nil)
      (let ((value (aux)))
        (if (not (= value 0))
            (setf result (append result (list value)))
            (return result))))))
```

a. (3p) Expliquer ce qui est calculé par la fonction `complicated` (étant donné que vous ne savez pas ce que fait la fonction `aux`).

b. (4p) Donner une liste de problèmes liés à cette définition.

c. (3p) Écrire une version améliorée de la fonction.

Exercice 2. (10p)

Dans une application de type Emacs, on souhaite écrire une macro permettant de définir une fonction *interactive* (c'est-à-dire, potentiellement appelée par l'utilisateur de l'application). Pour cela on souhaite rajouter à la syntaxe de `defun` la possibilité de préciser comment obtenir les arguments interactivement. On souhaite par exemple pouvoir écrire :

```
(defun-inter mail-file-to ((filename (minibuffer-read-file))
                           (to (minibuffer-read-string))
                           (signature (numeric-argument)))
  ...
  ...)
```

Le résultat d'un tel appel doit donner deux choses. D'abord une fonction `mail-file-to` doit être définie, comme si on avait écrit :

```
(defun mail-file-to (filename to signature)
  ...
  ...)
```

et deuxièmement, une entrée doit être définie sur la liste de propriétés du symbole `mail-file-to` contenant une fonction anonyme définie avec :

```
(setf (get 'mail-file-to 'interactive
           #'(lambda () (mail-file-to (minibuffer-read-file)
                                      (minibuffer-read-string)
                                      (numeric-argument)))))
```

a. (7p) Écrire une macro Common Lisp `defun-inter` autorisant la syntaxe suivante :

```
(defun-inter name ((arg1 exp1) ... (argn expn))
  bodyexp1
  ...
  bodyexpm)
```

qui donne le résultat souhaité.

b. (3p) Quelle expression permet d'appeler la fonction `mail-file-to` de façon à ce que ses arguments soient saisis *interactivement* (i.e., que doit faire l'équivalent de `M-x mail-file-to<RET>`)?