Université Bordeaux 1, Sciences et Technologies
UFR Mathématiques et Informatique

Maîtrise d'Informatique
Session de septembre 1999

Méthodes et Techniques de Programmation

Durée : deux heures.

Tous documents autorisés sauf photocopies 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. (5p)

Un *ensemble totalement ordonné* est un type de données dont les éléments sont totalement ordonnés par une relation d'ordre. Les opérations définies sur un ensemble totalement ordonné sont:

- création d'un ensemble vide,
- insertion d'un élément dans un ensemble,
- suppression d'un élément d'un ensemble,
- test si un élément est dans un ensemble,
- test si l'ensemble est vide,
- récupération de l'élément le plus petit (selon la relation d'ordre).
- union de deux ensembles,
- intersection de deux ensembles,

Écrire une implémentation en C du type *ensemble totalement ordonné*. L'interface doit être impérative. L'implémentation doit utiliser une liste chaînée triée par l'ordre entre les éléments.

Exercice 2. (5p)

Écrire une implémentation d'une pile en Java. L'interface doit être impérative. L'implémentation doit utiliser un tampon (un tableau d'éléments), qui doit être ré-alloué en fonction du nombre d'éléments dans la pile pour ne pas imposer une borne sur le nombre d'éléments. Vérifier qu'une suite de n opérations prenne $O(n)$ temps d'exécution.

Exercice 3. (5p)

Écrire une macro `bind-and` Common Lisp similaire à la forme spéciale `and` permettant d'écrire `(bind-and <expr1> <var> <expr2>)` où `<expr1>` et `<expr2>` sont des expression, et `<var>` est le nom d'une variable. Comme avec la forme spéciale `and`, `<expr1> est d'abord évaluée`. Si sa valeur est `nil`, alors la valeur de la forme entière est aussi `nil`. Sinon (si la valeur de `<expr1>` n'est pas `nil`), alors l'expression `<expr2>` est évaluée. Dans ce cas, la valeur de la forme entière est la valeur de l'expression `<expr2>`. La différence entre `bind-and` et `and` et que dans `bind-and`, pendant l'évaluation de `<expr2>`, une variable temporaire `<var>` contient la valeur de l'expression `<expr1>`. Ceci permet d'écrire des expressions comme `(bind-and (get jean 'mere) m (print m))`. Écrire une implémentation de la macro `bind-and`.

Exercice 4. (5p)

Étant donnée l'implémentation de `bind-and` de l'exercice 3, écrire une macro `bind-and*`, généralisation de `bind-and` à un nombre arbitraire de couples variable + expression permettant par exemple des expressions du type: `(bind-and* (get jean 'mere) m (get m 'pere) gp (progn (print m) (print gp)))`