Université BORDEAUX I

Année 1992-93
Session de septembre

Maîtrise d'informatique

Modèles et Techniques de Programmation (J.P.Braquelaire)

Durée: 2 heures
Tous documents autorisés

Première partie

Le but de ce problème est de définir un mécanisme de mémoire en programmation fonctionnelle pure. Une mémoire M est un ensemble de couples $(\langle symbole \rangle, \langle valeur \rangle)$. Elle est implémenté au moyen d'une **fonction** qui reçoit en argument un symbole s et retourne

- la valeur v si $(s, v) \in M$,
- `nil` sinon.

L'implémentation doit être effectuée en programmation fonctionnelle pure (sans effet de bord), et sans utiliser de liste.

1) – 1 point – Écrire en SCHEME la fonctionnelle `memoire-vide` qui crée une mémoire vide.

2) – 4 points – Écrire en SCHEME la fonctionnelle `ajouter` qui reçoit trois paramètres : une mémoire M (c'est-à-dire une fonction), un symbole S et une valeur V, et qui retourne la mémoire M' codant $M \cup (s, v)$.

3) – 3 points – Calculer la valeur de l'expression SCHEME

```
(ajouter
 (ajouter
  (ajouter (memoire-vide) 'un 1)
  'deux 2)
 'trois 3)
```

On donnera une λ-expression, en utilisant soit la syntaxe du λ-calcul, soit une syntaxe équivalente en SCHEME. Dans le premier cas, on supposera on supposera définis les opérateurs ***si*** et ***égale***, les entiers, et les symboles.

4) – 2 points – On associe la mémoire de la question précédente au symbole `m`:

```
(define m (ajouter (ajouter ...) 'trois 3))
```

et on évalue l'expression

```
(define m (ajouter m 'm m))
```

Donner l'expression d'accès à `m` retournant la valeur `2`. Expliquer comment elle est évaluée.

Deuxième partie

On appellera *roue numérique* un générateur parcourant indéfiniment une liste d'entiers de longueur finie, en revenant sur le premier élément de la liste chaque fois que le dernier est atteint. Une roue numérique est définie par un couple d'entiers qui sont les deux valeurs extrêmes de la liste qu'elle génère. Ainsi la roue numérique définie par le couple $\prec 1, 5 \succ$ produit la suite infinie

`1 2 3 4 5 1 2 3 4 5 1 2 ...`

Le but de ce problème est d'étudier une implémentation par objet de ce module en langage C standard.

1) – 1 point – Définir la structure `struct roue_num` codant l'état d'un objet de la classe `roue_num`.

2) – 4 points – De manière générale, on implémente l'envoi d'un message $\boldsymbol{m}$ à un objet $\boldsymbol{o}$ de la classe ***cls*** par un appel à la fonction ***cls_m***. Plus précisément, l'envoi à $\boldsymbol{o}$ du message $\boldsymbol{m}$ avec les paramètres $\boldsymbol{e_1}$, ..., $\boldsymbol{e_k}$, correspond à l'appel de fonction

$$\boldsymbol{cls\_m(o,\ e_1,\ \ldots,\ e_k)}$$

Il est possible d'envoyer trois messages à une roue numérique :

- `initialiser` : qui repositionne la roue au début;
- `avancer` : qui fait avancer la roue d'une position (cette fonction retourne vrai lorsque la roue repasse par zéro et faux sinon);
- `valeur` : qui retourne la valeur courante de la roue.

Tous ces messages sont sans paramètre. Donner l'implémentation des trois méthodes correspondantes.

3) – 3 points – L'interface de la classe est placée dans un fichier `roue_num.h` et son implémentation dans un fichier `roue_num.c`. Afin de garantir le masquage de l'implémentation, on ne veut pas exporter la définition de la structure `struct roue_num` hors du fichier `roue_num.c`. Proposer une organisation des fichiers `roue_num.h` et `roue_num.c` qui satisfasse cette condition. Le fichier `roue_num.h` devra contenir la définition du type d'un objet de la classe `roue_num` au moyen de la construction `typedef`.

4) – 2 points – Comment effectuer l'instanciation et la libération d'un objet de cette classe ?