Master STL - M1 | **Compilation Avancée** | **Université Pierre et Marie Curie**
Module CA (MI190) | *Documents autorisés* | **Année 2006–2007**

Examen du 12 juin 2007

1ère partie (10 points) - à traiter sur une copie séparée

1ère partie : implantation d'un récupérateur automatique de mémoire

On cherche à implanter une version simplifiée de l'algorithme de Mark&Compact en le décomposant en 4 passes :

1. phase de marquage
2. phase de compactage découpée en 3 sous-phases :
   (a) calcul des adresses, pour connaître où vont être déplacées les cellules
   (b) mise à jour des pointeurs, à l'intérieur des cellules par rapport au déplacement prévu
   (c) déplacement des objets

On demande d'écrire cet algorithme dans un pseudo-langage (pseudo-C, pseudo-Caml, pseudo-Java). et à l'illustrer sur le petit exemple suivant :

```
let l  =  ['b';'c';'d'] ;;
let a  =  match ( ['a'] , l )
          with  p  ->  ( fst p , snd p ) ;;
```

Ce code alloue deux listes `['b';'c';'d']` et `['a']` dans le tas, suivi d'une paire pour construire la paire de ces deux listes, puis une deuxième paire est créée pour relier les deux listes. Au final, on trouve dans l'ensemble des racines la première liste (variable `l`) et la deuxième paire (variable `a`). La figure 1 montre ces allocations.

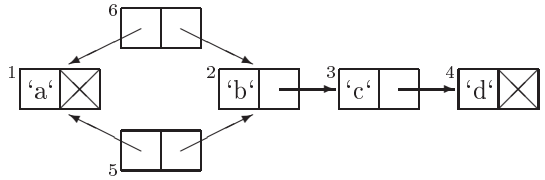

Figure 1: la représentation d'une valeur O'Caml

Et la figure 2 ces mêmes allocations avec le tas représenté par un vecteur de cellules de deux cases.

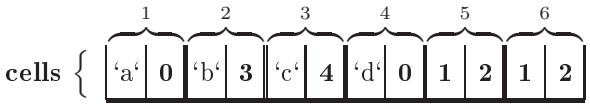

Figure 2: Tas avant Garbage Collecting

Pour simplifier l'implantation on suppose que le tas ne contient que des cellules regroupant 2 valeurs.

1. Définir dans le langage d'implantation de votre choix le type `element` pouvant représenter soit un caractère, soit un entier (ou une adresse), le type `cell` pour les paires d'éléments auxquelles on ajoute deux autres champs : le premier pour indiquer le marquage et le deuxième pour contenir l'index (ou l'adresse) du futur déplacement, et le type `tas` contenant un tableau de `cell` et un entier correspondant à la longueur maximale du tas et un entier indiquant l'index de la première cellule libre. Ecrire la valeur `montas` de type `tas` correspondant à la figure 2.

2. Ecrire la phase de marquage des cellules du tas. Commencer par écrire les fonctions ou macros permettant de marquer (`MARQUE`) une cellule, de vérifier (`EST_MARQUEE`) si une cellule est marquée et de démarquer (`DEMARQUE`) une cellule. Ecrire le déclenchement de cette phase sur la valeur `montas`.

3. On s'intéresse à la phase de calcul des adresses. On pourra supposer connue les fonctions ou macros suivantes : `EST_ADR` prenant un élément et indiquant si c'est une adresse ou bien un caractère, et `LIBRECELL` prenant un tas et un index et retournant l'index de la première cellule non marquée à partir de cet index. Ecrire la phase de calcul d'adresse en parcourant les cellules marquées et en indiquant dans la cellule vers quelle cellule non marquée (du début du tas) elle devra être déplacée.

4. La phase suivante consiste à mettre à jour les pointeurs (ou index) dans les cellules en tenant compte du futur déplacement de la cellule pointée (ou indexée). Ecrire cette phase.

5. La dernière phase effectue effectivement le déplacement des cellules marquées (en les démarquant) vers les cellules libres prévues. Ecrire cete phase.

6. Donner une représentation graphique du tas en fonction de la structure de donnée de la question 1 et simuler graphiquement l'enchaînement des quatre phases en fonction de votre version du programme de GC.