Master STL - M1 | **Compilation Avancée** | **Université Pierre et Marie Curie**
Module CA (4I504) | *Documents autorisés* | **Année 2017–2018**

Examen du 26 juin 2018

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&Sweep en le décomposant en 2 passes :

1. phase de marquage
2. phase de récupération qui balaie l'ensemble du tas en effectuant 2 actions :
   (a) démarquage des cellules à conserver
   (b) construction de la liste des cellules libres

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.

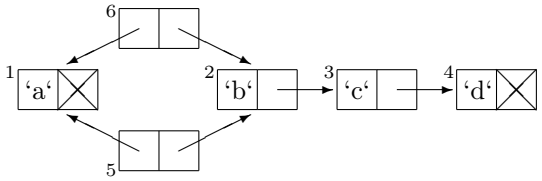

Figure 1: la représentation d'une valeur O'Caml

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

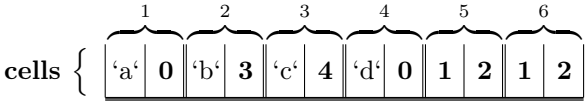

Figure 2: Tas avant Garbage Collecting

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 un autre champ pour indiquer le marquage, 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. L'index de la première cellule libre correspond à la liste des cellules libres. Ecrire la valeur `montas` de type `tas` correspondant à la figure 2 sans oublier d'initialiser la liste des cellules libres.

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 récupération. Préciser bien la valeur de la liste des cellules libres au début de la phase de récupération puis écrire la phase de récupération.

4. Donner une représentation graphique du tas (des figures 1 et 2) en fonction de la structure de données de la question 1 et simuler graphiquement l'enchaînement des différentes phases en fonction de votre version du programme de GC.

5. On cherche à rendre cet algorithme incrémental, c'est-à-dire de ne pas effectuer toute la phase de récupération en une seule étape mais petit à petit. L'idée est de ne pas bloquer le programme par une phase de récupération trop longue. Pour cela on arrête la phase de récupération dès que l'on a une liste de cellules libres d'une certaine taille donnée, appelée `TAILLE_RECUP`. Au prochain GC, on continue la récupération. Si elle se termine avant d'avoir récupéré une liste de cellules libres de la bonne taille, alors on relance la phase de marquage suivie d'une phase de récupération incrémentale. Dans le cas où l'on ne peut pas récupérer la taille donnée, alors le GC échoue. Indiquer comment modifier votre programme pour tenir compte de ces changements, puis implanter ces modifications.

6. On s'intéresse maintenant aux programmes concurrents multi-threadés. Que faut-il ajouter à votre programme si plusieurs threads cherchent à allouer "en même temps" dans la même zone mémoire ? Une fois le GC enclenché, si un autre thread cherche à allouer, il se met en attente de la fin du GC pour pouvoir effectuer cette action. Indiquer ce que vous devez ajouter à votre algorithme pour garantir la correction de celui-ci dans un cadre multi-threadé (seul le programme est multi-threadé, pas le GC). Implanter vos modifications.

7. Dans le cadre des questions précédentes, une fois une zone de `TAILLE_RECUP` récupérée, la fin de la récupération peut être elle-même lancée en concurrence via un thread. Les choses se compliquent un peu si ce dernier thread n'a pas fini alors qu'un autre thread nécessite une allocation mais qu'il n'y a plus assez d'espace. Indiquer les modifications à apporter à votre programme et les implanter.