Université Laval
Faculté des sciences et de génie
Département d'informatique et de génie logiciel
GLO-66811

Danny Dubé
Automne 2008
Version : 14 septembre

Travail pratique #1
Programmation en Scheme

Ce travail pratique consiste à vous faire pratiquer la programmation en Scheme. La tâche à accomplir n'a pas de lien spécifique avec le domaine de la compilation mais vise simplement à vous exercer en programmation fonctionnelle. Votre travail consiste à écrire un programme qui cherche la solution à un jeu mathématique.

La section suivante présente le jeu à solutionner ainsi que les fonctions que vous devez programmer. Ensuite, quelques directives pédagogiques sont données.

Réorganisation d'entrepôt

La tâche consiste à déplacer des caisses dans un entrepôt pour passer d'une disposition à une autre. Pour fins de simplification, on suppose que l'entrepôt est fait en long ; c'est-à-dire que toutes les caisses peuvent être accumulées le long d'une seule rangée. Les caisses peuvent être empilées les unes sur les autres aussi haut que l'on veut. Les (piles de) caisses ne peuvent se trouver qu'à des positions bien spécifiques, numérotées 0, 1, ..., $n-1$. Ceci signifie que l'on ne peut placer une caisse sur le sol hors de ces positions, sur une autre caisse en porte-à-faux ou déposée sur deux autres caisses à la fois. Une pince manipulatrice permet de déplacer une caisse à la fois. Elle prend la caisse sur le dessus d'une pile dite *de départ* et la déplace sur le dessus d'une pile dite *d'arrivée* (ou directement sur le sol s'il n'y avait aucune caisse présente à la position d'arrivée).

La réorganisation est l'opération qui consiste à planifier une séquence de mouvement de la pile qui permet de passer d'une disposition à l'autre. Chaque mouvement est une paire de positions : la position d'où une caisse est prise et la position où la caisse est déposée.

Vous devez écrire un programme qui comporte les fonctions `reorg-s` et `reorg-m` (pour *simple* et *multiple*) qui permettent de planifier des réorganisations. Naturellement, d'autres fonctions devraient apparaître dans le programme. Chacune de ces deux fonctions prend deux dispositions en arguments, soit la disposition initiale et la disposition voulue, et retourne une séquence de mouvements. Chaque disposition est une liste de longueur n contenant la description des n piles dans l'entrepôt. Une description de pile est une liste contenant le nom de chaque caisse dans la pile, en commençant par la caisse du dessus. Chaque caisse est identifiée par un symbole. Une séquence de mouvements est une liste contenant des paires de positions.

La fonction `reorg-s` permet de réorganiser un entrepôt où toutes les caisses sont identifiées par un nom unique. La fonction `reorg-m` permet de réorganiser un entrepôt où plusieurs caisses peuvent avoir le même nom. On considère deux caisses ayant le même nom comme étant indistinguables. Voici des exemples d'utilisation des fonctions de réorganisation :

```
> (reorg-s '((a) (b) (c)) '((b a) (c) ()))
((1 . 0) (2 . 1))
> (reorg-m '((a b) () (a b)) '((b a) () (b a)))
((0 . 2) (0 . 1) (2 . 0) (1 . 0) (2 . 0) (2 . 1) (0 . 2) (1 . 2))
```

Les fonctions peuvent être programmées en supposant que les arguments sont valides et qu'ils admettent l'existence d'une solution. En particulier : l'entrepôt ne change pas de taille ; on retrouve les mêmes caisses dans les deux dispositions ; il existe une façon de déplacer les caisses pour passer de la première disposition à la seconde. Faites attention à ne pas supposer qu'il y a une taille minimale pour l'entrepôt : un entrepôt à deux ou une positions est possible et même un entrepôt à zéro position. Il ne s'agit pas d'un problème d'optimisation où on cherche la plus courte séquence de mouvements. En effet, le deuxième exemple ci-haut admet des solutions plus courtes. Bien entendu, aucune des fonctions ne peut boucler à l'infini (lorsque le problème à solutionner est valide).

Votre programme doit être écrit dans un style purement fonctionnel, c'est-à-dire il ne doit pas comporter d'affectations à des variables ni de mutations de structures de données. Notez que les résultats présentés dans les deux exemples sont affichés non pas par votre programme mais par l'interprète lui-même. Vos fonctions `reorg-s` et `reorg-m` doivent se contenter de *retourner* leur réponse. Je vous suggère de programmer plusieurs petites fonctions plutôt que d'agglomérer toute la logique dans quelques fonctions géantes.

Remise des travaux

Vous devez remettre votre programme **via l'intranet**. Vous devez aussi remettre un rapport qui explique vos choix d'implantation et qui présente la façon de solutionner le problème. Le rapport est très important ; il vaut environ la moitié des points de ce travail. Programmer la fonction `reorg-s` sans programmer `reorg-m` est considéré comme l'équivalent de 80% du travail.

Vous devez remettre le devoir au plus tard le **26 septembre, 18h00**. Le travail doit être fait individuellement. Des discussions verbales sont acceptables entre camarades mais pas des échanges de bouts de réponses. De plus, il va de soi que vous ne devez pas récupérer sur internet la solution partielle ou complète à ce travail pratique.