Master Ingénierie Informatique — 2009–2010
Module *Programmation logique et par contraintes avancée* — Feuille de Examen

Examen du cours *Programmation logique et par contraintes avancée*

A côté de chaque exercice on donne, à titre indicatif, un barème.

Documents autorisés : transparents du cours, énoncés et corrigés de TP, notes personnelles. Des ordinateurs et des livres ne sont pas autorisés. Les téléphones portables, comme tout autre moyen de communication vers l'extérieur, doivent être éteints. Le temps à disposition est de 2 heures.

Exercice 1 (7 points)

On cherche à payer une somme d'argent en espèces, mais en utilisant un nombre minimal de pièces. Les pièces disponibles sont décrites par un enregistrement, dont les champs sont les dénominations possibles, et la valeur au champs d indique le nombre de pièces disponibles de dénomination d. Par exemple,

```
bac(100:6  25:8  10:10  5:1  1:5)
```

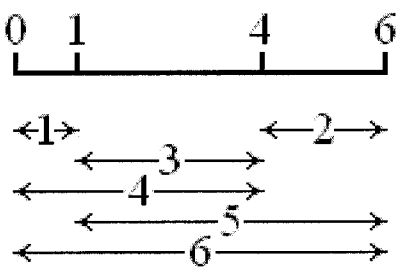
exprime qu'on a 6 pièces à 100 centimes, 8 pièces à 25 centimes, 10 pièces à 10 centimes, 1 pièce à 5 centimes et 5 pièces à 1 centime.

Écrire une fonction Oz qui prend en argument une description des pièces disponibles comme au dessus, et un entier non négatif qui est la somme à payer. Le résultat de la fonction est un enregistrement qui associe à chaque dénomination le nombre de pièces à payer, ou nil s'il n'y a pas de solution. On veux que le nombre total de pièces soit minimal.

Votre solution doit fonctionner pour n'importe quel choix de dénominations (par exemple on pourrait avoir en plus des pièces à 2 centimes).

Exercice 2 (13 points)

Une règle de Golomb, appelée ainsi en l'honneur du mathématicien Solomon W. Golomb, est une règle munie de marques à des positions entières, telle que deux paires de marques ne soient jamais à la même distance ; en d'autres termes, chaque couple de marques mesure une longueur différente des autres. Par définition, l'ordre d'une règle de Golomb est le nombre de marques qu'elle porte ; la longueur d'une règle de Golomb est la plus grande distance entre deux de ses marques. Voici par exemple une règle de Golomb d'ordre 4 avec longueur 6 :



Il n'est pas nécessaire qu'une règle de Golomb permette de mesurer toutes les distances entières entre 0 et la longueur de la règle mais si c'est le cas, on dit qu'il s'agit d'une règle de Golomb parfaite. La règle de l'exemple ci-dessus est parfaite.

Écrire une fonction Oz qui prend en argument un entier positif N, et qui envoie une règle de Golomb optimale d'ordre N. Utiliser les contraintes à domaine fini, et une recherche en deux dimensions. Est-ce qu'il y a des symétries qu'on peut exploiter ? Expliquez aussi comment modifier votre solution pour chercher seulement des règles de Golomb parfaites.

Par exemple, lancée sur l'argument 4 la fonction pourrait renvoyer le résultat `[0 1 4 6]`.

Quelques fonctions utiles de la librairie

`{FD.sum Dv '=:' D}`, où Dv est un vecteur de variables $D_1, \ldots, D_n$ à domaine fini et D une variable à domaine fini, est un propagateur pour la contrainte

$$D_1 + D_2 + \ldots + D_n = D$$

`{FD.sumC Iv Dv '=:' D}`, où Iv est un vecteur de valeurs *déterminées* $I_1, \ldots, I_n$ et Dv est un vecteur de variables $D_1, \ldots, D_n$ à domaine fini et D une variable à domaine fini, est un propagateur pour la contrainte

$$I_1 * D_1 + I_2 * D_2 + \ldots + I_n * D_n = D$$

On peut aussi utiliser, dans les deux cas, des relations `\=:`, `<:`, etc. à la place de `=:`.