TP 3 - PLC avec Swi-Prolog - S4

Ajouter `:- use_module(library(clpfd)).` en tête de votre programme.

Exercice 1 (**Pour s'échauffer**)

On utilise 4 variables uniquement : Xi est la ligne de la reine placée en colonne i (voir le cours de la semaine dernière à l'adresse suivante `http://web4.ensiie.fr/~dubois/IA/IA/ia_csp.pdf`).
Ecrivez le programme correspondant à la 2ème modélisation des 4 reines.
Combien y-a-t-il de solutions ?

Exercice 2 (**Retour de monnaie**)

On s'intéresse à un distributeur automatique de boissons. L'utilisateur insère des pièces de monnaie pour un total de T centimes d'Euros, puis il sélectionne une boisson, dont le prix est de P centimes d'Euros (T et P étant des multiples de 10). Il s'agit alors de calculer la monnaie à rendre, sachant que le distributeur a en réserve E2 pièces de 2 euros, E1 pièces de 1 euro, C50 pièces de 50 centimes, C20 pièces de 20 centimes et C10 pièces de 10 centimes.

1. **Modélisez ce problème sous la forme d'un CSP.**
   Indications : Pour modéliser ce problème sous la forme d'un CSP, il s'agit d'identifier les variables (les inconnues du problème), les domaines de valeur de ces variables, et les contraintes existant entre ces variables. Ici, T, P, E2, E1, C50, C20 et C10 sont des "données" du problème (correspondant aux paramètres en entrée).
   Ce que l'on doit déterminer (nos inconnues), c'est la quantité de pièces de 2 et 1 Euro, ainsi que de 50, 20 et 10 centimes à rendre. On a donc 5 variables. Pour modéliser notre problème sous la forme d'un CSP (X,D,C), vous devez
   donner un nom à chacune de ces variables, et définir X comme étant l'ensemble de ces 5 variables ;
   définir pour chacune de ces 5 variables son domaine de valeur, sachant que la quantité de pièces retournées, pour un type de pièce donné, est comprise entre 0 et le nombre de pièces de ce type que l'on a en réserve ;
   définir les contraintes (il n'y en a qu'une... elle spécifie que la somme à retourner doit être égale à la somme insérée moins le prix à payer).
2. Le programmer en Swi-Prolog : écrivez le prédicat monnaie/4 suivant :
   monnaie(TotalDonné, TotalDu, Pièces_en_réserve, Pièces_à_retourner) réussit si
   TotalDonné est la somme insérée dans le distributeur (en centimes d'euros),
   TotalDu est la somme à payer (en centimes d'euros),
   Pièces_en_réserve est la liste des quantités de pièces en réserve dans le distributeur,
   Pièces_à_retourner s'unifie avec la liste des quantités de pièces à rendre par le distributeur.
   La façon la plus simple à programmer consiste à décider que les deux listes de pièces sont des listes d'entiers ordonnés en fonction des valeurs des pièces. Par exemple, on mettra toujours la quantité de pièces de 2 Euros en premier, puis celles de 1 Euro, puis celles de 50 centimes, 20 centimes et 10 centimes. La requête sera alors ; `monnaie1(600,530,[5,7,4,3,5],L).`

Exercice 3 (**Send more money**)

On considère l'addition suivante :

```
  SEND
+ MORE
-------
= MONEY
```

où chaque lettre représente un chiffre différent (compris entre 0 et 9). On souhaite connaitre la valeur de chaque lettre, sachant que la première lettre de chaque mot représente un chiffre différent de 0.

1. Formaliser le problème sous la forme d'un CSP (X,D,C)
2. Programmer le prédicat send/1 en Swi-prolog qui prend en paramètres les différentes lettres. Il doit calculer la ou les solutions.

Exercice 4 (**Des commandes de pizzas**)

Claude, Jean, Marie, et Yves veulent faire une commande groupée pour des pizzas. Il y a quatre types de pizza : Margherita, Picante, Romaine, Hawai. Une pizza se coupe en quatre morceaux exactement.

Ecrire en Swi-Prolog, et en utilisant les contraintes à domaines finis, un prédicat pizza(Margherita,Picante,Romaine,Hawai) qui calcule le nombre de pizzas de chaque sorte à commander, tel que toutes les préférences ci-dessous sont satisfaites, et tel qu'aucun morceau ne reste à la fin.

1. Claude veut au total au moins 3 morceaux, et au plus 5 morceaux. Il n'aime pas du tout la pizza Romaine, et il veut au plus un morceau de pizza Hawai.
2. Jean veut 4 morceaux de pizza, dont au moins deux morceaux de Picante.
3. Marie veut au total 3 morceaux de pizza. Elle veut manger autant de morceaux de Picante qu' Yves.
4. Yves veut manger au moins 6 morceaux de pizza, au plus 10, dont au plus la moitié de pizza Romaine.

Donnez une solution.
Puis comptez le nombre de solutions différentes.

Indication : la modélisation utilisera les variables CM, CP, CR, CH, JM, JP etc qui indiquent, pour chaque personne, le nombre de morceaux de pizza de chaque sorte (la première lettre indique la personne, la deuxième la sorte de pizza).

Exercice 5 (**Problème 7-11**)

A guy walks into a 7-11 store and selects four items to buy. The clerk at the counter informs the gentleman that the total cost of the four items is $7.11. He was completely surprised that the cost was the same as the name of the store. The clerk informed the man that he simply multiplied the cost of each item and arrived at the total. The customer calmly informed the clerk that the items should be added and not multiplied. The clerk then added the items together and informed the customer that the total was still exactly $7.11.

What are the exact costs of each item ? What is the number of solutions ?

Indication : travailler en nombres entiers.

Exercice 6 (**Emploi du temps**)

Les organisateurs d'un congrès ont 3 salles et deux jours pour 11 sessions de demi-journées (A,B,C,...,K). Les ensembles de sessions AJ, JI, IE, CF, FG, DH, BD, KE, BIHG, AGE, BHK, ABCH, DFJ ne peuvent pas se dérouler en même temps (il existe au moins un participant dans toutes les sessions de ces ensembles). De plus la session E doit se dérouler avant la session J, et les sessions D et F avant K. Les organisateurs souhaitent déterminer un emploi du temps. Ecrire le programme Swi-Prolog permettant de l'établir.

Exercice 7 (**Carrés latins**)

Généraliser le problème des carrés latins de manière à réousdre des problème de taille n (matrice n x n).

Exercice 8 (**Sudoku**)

Le problème du Sudoku consiste à remplir la grille de sorte que chaque ligne, chaque colonne et chaque carré contiennent les chiffres 1 à 9. Par exemple :

		9			1	6	2	
5	7			2	8		3	
3			7					4
8	9			7		4		
	6		5		3		9	
		1		9			7	6
6					7			8
	4		1	3			6	5
	2	7	6			9		

Ecrire un programme en Swi Prolog pour résoudre des grilles de sudoku. Vous écrirez un prédicat sudoku(L) qui réussit quand le sudoku avec les valeurs initiales données par L a une solution. La liste L est une liste de triplés (ligne, colonne, valeur). Ci-dessous la définition de la liste de l'exemple donnée précédemment :

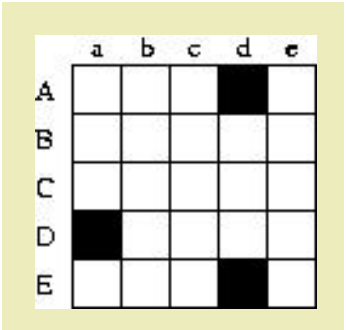
```
[(1,3,9), (1,6,1), (1,7,6), (1,8,2),
(2,1,5), (2,2,7), (2,5,2), (2,6,8), (2,8,3),
(3,1,3), (3,4,7), (3,9,4),
(4,1,8), (4,2,9), (4,5,7), (4,7,4),
(5,2,6), (5,4,5), (5,6,3), (5,8,9),
(6,3,1), (6,5,9), (6,8,7), (6,9,6),
(7,1,6), (7,6,7), (7,9,8),
(8,2,4), (8,4,1), (8,5,3), (8,8,6), (8,9,5),
(9,2,2), (9,3,7), (9,4,6), (9,7,9)]
```

Vous pourrez utiliser le prédicat displayline pour afficher la solution trouvée.

```
displayline(L) :- printf('|%d %d %d|%d %d %d|%d %d %d|%n',L).
displayseperator :- printf('+-----+-----+-----+%n',[]).
```

Exercice 9 (**Nombres croisés**)

Ecrire un programme résolvant la grille de nombres croisés ci-dessous. Vous représenterez la grille par la liste des cases blanches.



Horizontalement :
A- Le produit des chiffres vaut 3.
B- La somme des chiffres vaut 12.
C- Carré de d.
D- C'est un carré.

E- Le produit des chiffres vaut 18.

Verticalement :
a- Le produit des chiffres vaut 2.
b- C'est un palindrome.
c- Carré de E.
d- Le produit des chiffres vaut 12.
e- La somme dees chiffres est un diviseur premier de E.

Les indications ci-dessus concernent les nombres (qui se lisent de gauche à droite ou de haut en bas) avant les cases noires. Les nombres ne commencent pas par 0.