Examen de programmation logique

6 juin 2017

La clarté et la précision de la rédaction seront prises en compte dans l'évaluation.
Le barème est indicatif.
Aucun document autorisé.
Machines (ordinateurs, téléphones, montres connectées et calculatrices) interdites.

Aucune question ne pourra être posée durant l'examen. En cas de doute concernant le sujet, vous expliquerez vos hypothèses.

Durée : 2h

1 Mise en jambes /4 pt

Écrire les 2 prédicats suivants :

1. `intervalle_decroissant` qui retourne vrai si la liste en sortie (troisième paramètre) contient les entiers compris entre `Binf` (premier paramètre) et `Bsup` (deuxième paramètre) dans l'ordre décroissant.

   ```
   ?- intervalle_decroissant(3,6,L).
   L = [6,5,4,3]
   ```

2. `coupe` qui retourne vrai si la liste `P` en sortie (troisième paramètre) est le préfixe de taille `N` (deuxième paramètre) et la liste `S` en sortie (quatrième paramètre) est le suffixe correspondant d'une liste en entrée (premier paramètre).

   ```
   ?- coupe([1,3,6,3,2],2,P,S).
   P = [1,3]
   S = [6,3,2]
   ```

2 Nombres parfaits /6 pt

On appelle nombre parfait un entier strictement positif qui est la somme de ses diviseurs propres. Par exemple, 6 et 28 sont parfaits puisque $6 = 1 + 2 + 3$ et $28 = 1 + 2 + 4 + 7 + 14$.

1. Écrire un prédicat `liste_diviseurs(N,ListeDiviseurs)` qui est vrai si `ListeDiviseurs` est la liste ordonnée des diviseurs de `N`.

   ```
   ?- liste_diviseurs(6,L).
   L = [1,2,3,6]
   ```

2. Écrire un prédicat `est_parfait(N)` qui est vrai si `N` est parfait.

   ```
   ?- est_parfait(6).
   true
   ```

3. Écrire un prédicat `liste_parfaits(Borne,L)` qui est vrai si `L` est la liste des nombres parfaits de `1` à `Borne`.

   ```
   ?- liste_parfaits(1000,L).
   L = [6,28,496]
   ```

3 Coloration /4 pt

Écrire un prédicat `coloration_optimale` qui retourne vrai si la liste en sortie (second paramètre non instancié) représente une coloration d'un graphe de `N` sommets numérotés de `1` à `N` (premier paramètre) avec le minimum de couleurs. Deux sommets adjacents ne peuvent pas avoir la même couleur. On considèrera que le graphe est modélisé par le prédicat `arete(S1,S2)`, indiquant qu'il y a une arête entre les sommets `S1` et `S2`. Les couleurs seront codées par des entiers consécutifs commençant à 1.

```
?- listing(arete),coloration_optimale(4,C).
arete(1, 2).
arete(1, 3).
arete(1, 4).
arete(2, 3).
C = [1, 2, 3, 2]
```

4 Rangement de boîtes /6pt

On veut ranger un ensemble de boîtes rectangulaires dans une boîte plus grande. Le problème est en 2D. Toutes les boîtes sont des rectangles de dimension entières. On suppose donnés un prédicat `boite/2` et un prédicat `grande_boite/2`. Par exemple, `boite(b1,dim(50,30)))` signifie qu'il existe une boîte de dimension 50*30.

1. Donner la syntaxe pour appeler la librairie adéquate.

   Pour décrire une boîte de dimensions `Longueur` et `Hauteur` dont le coin inférieur gauche est à une abscisse `X` et une ordonnée `Y`, on utilisera le prédicat `rect((X,Y),dim(Longueur,Hauteur))`.
2. Écrire un prédicat `rotation(Boite1,Boite2)` qui donne les rotations possibles d'une boîte en gardant les côtés parallèles à ceux de la grande boîte.
3. Écrire un prédicat `contenu_dans` qui est vrai si une boîte de dimensions `Longueur` et `Hauteur` dont le coin inférieur gauche est à une abscisse `X` et une ordonnée `Y` est entièrement contenu dans une grande boîte de dimensions `LongueurGB` et `HauteurBG` dont le coin inférieur gauche est à une abscisse `XGB` et une ordonnée `YGB`.
4. Écrire un prédicat `pas_de_chevauchement(Boite1,Boite2)` qui est vrai si deux boîtes ne se chevauchent pas.
5. Écrire un prédicat `rangement(GrandeBoite,ListeBoites)` qui est vrai si l'ensemble des boîtes de `ListeBoites` peuvent être positionnées dans `GrandeBoite` sans chevauchement.

Bonus

Quel est le nom de l'inventeur de Prolog disparu le 12 mai dernier ?

♣♢♡♠