Université d'Orléans **M1 Informatique**
Programmation en logique et par contraintes **Année 2010-2011**

Contrôle continu du 5 novembre 2010

Durée : 2h. Documents autorisés : notes de cours et de TD. Expliquez et commentez bien vos programmes. Le sujet est sur 2 pages.

Exercice 1 Le prédicat *insere(+X,+L, ?R)* est définit comme suit :

```
insere(X,[],[X]).
insere(X,[Y|L],[X,Y|L]) :-
    X =< Y.
insere(X,[Y|L],[Y|R]) :-
    insere(X,L,R).
```

1. Donner l'arbre de dérivation ainsi que les réponses pour le but suivant :

   ```
   ?- insere(2,[1,3,5],L).
   ```

2. Quel est l'objectif de ce prédicat ?
3. On veut que pour que l'insertion de X dans une liste déjà triée L donne une liste triée R. Comment modifier ce prédicat avec une coupure ? Donnez ensuite pour le prédicat modifié l'arbre de dérivation ainsi que les réponses pour le but de la question 1.

Exercice 2

1. Écrivez un prédicat *compter(+L,+X,-N)* pour compter le nombre de fois qu'un élément X apparaît dans la liste L.
2. Une suite $[x_0, x_1, \ldots, x_{n-1}]$ de longueur n est appelée une suite magique si chaque élément x_i est le nombre de fois que i apparaît dans la suite. Par exemple pour la longueur $n = 4$ il y a deux suites magiques $[2, 0, 2, 0]$ et $[1, 2, 1, 0]$. En utilisant le prédicat *compter* défini précédemment, écrivez un prédicat *suiteMagique(+L)* qui réussit si L représente une suite magique.
3. Ecrivez un prédicat *dk(+K,+L, ?X)* qui vérifie (ou calcule) si l'élément X apparaît dans la liste L à toutes les positions qui sont un multiple de K. Par exemple

   ```
   ?- dk(3,[a,b,c,d,e,c,f],c).
   yes
   ?- dk(3,[a,b,c,d,e,c,f],X).
   X=c
   ```

Exercice 3 Mini-Scrabble

On modélise une petite portion du jeu de "Scrabble" en Prolog. Pour cela, on dispose du prédicat *valeur(L,N)* qui est vrai si la lettre L a la valeur N. On suppose donc que toutes les lettres de l'alphabet ont leur valeur définie de la manière suivante :

```
valeur(a,1).
valeur(b,3).
valeur(c,3).
etc.
```

Un "mot" que l'on souhaite poser sur le plateau est modélisé par la liste de ses lettres. Chaque question peut être traitée indépendamment en supposant faites les questions précédentes.

1. Dans un premier temps, on suppose que l'on a dans une liste de nombres entiers la suite des valeurs des cases sur lesquelles on pose le "mot". Une case "normale" a la valeur 1, une case "lettre compte double" la valeur 2 et "lettre compte triple" la valeur 3. Pour simplifier le problème, on ne prend pas en compte les cases "mot compte double (ou triple)". Écrire le prédicat *points(M,C,V)* où M est la liste de lettres contenant le mot que l'on pose, C est la liste de même taille des valeurs des cases où on le pose et V, que l'on veut calculer, est la valeur du résultat. Par exemple, en mode interrogation :

```
?- points([b,a,c],[1,1,1],V).
V=7
?- points([b,a,c],[1,3,2],V).
V=12
```

2. On suppose maintenant que la liste C des valeurs des cases contient exactement 15 chiffres (la taille d'une ligne complète du plateau de jeu). Ecrire un nouveau prédicat *points_ligne(M,C,N,V)* dans lequel le nouvel argument N indique la position à partir de laquelle on "pose" la première lettre du mot. On ne demande pas de vérifier que le mot ne "déborde pas" du plateau. On aura par exemple :

```
?- points_ligne([b,a,c],[3,1,1,2,1,1,1,3,1,1,1,2,1,1,3],1,V).
V=13
?- points_ligne([b,a,c],[3,1,1,2,1,1,1,3,1,1,1,2,1,1,3],2,V).
V=10
```

3. Définir un prédicat *meilleur_coup(M,C,N,V)* qui, cette fois, pour un mot donné M et une suite de 15 cases C, donne la première position N de la première lettre du mot qui donnera le meilleur résultat V. On suppose connu le prédicat *compte(L,T)* qui est vrai si la liste L est de taille T. On aura par exemple :

```
?- meilleur_coup([b,a,c],[3,1,1,2,1,1,1,3,1,1,1,2,1,1,3],N,V).
N=1
V=13
```