UQÀM Université du Québec à Montréal

Paradigme de programmation (inf-2160)
Hiver 2012

Examen intra
7 mars 2012

CONSIGNES

· **Les règlements de l'UQAM concernant le plagiat seront strictement appliqués.**

· Il est important de bien expliquer vos choix s'il y a lieu.

· La documentation est permise.

· La durée de l'examen est de 3 heures.

· Vous pouvez utiliser les versos comme brouillon ou comme espace supplémentaire.

· **Il est interdit de dégrafer le questionnaire.**

· Les téléphones cellulaires, calculatrices, ordinateurs, palm, baladeurs, iPods, etc. sont interdits.

IDENTIFICATION

NOM : _____

PRÉNOM : _____

CODE PERMANENT : _____

SIGNATURE : _____

GROUPE : _____

PROFESSEUR : _____

#1 _____ / 12

#2 _____ / 13

#3 _____ / 13

#4 _____ / 12

TOTAL

_____ / 50

commentaire :

Numéro 1. (12 pts)
Objectif(s) :

- **Application des connaissances.**
- **Lecture de code Prolog.**
- **Coupure.**
- **Dynamic, asserta et retract.**

Question :

```
:- dynamic a/2.

defaire( [] ).
defaire( [ X | XS ] ) :-
        asserta( a( X ) ),
        defaire( XS ).

faire( [] ) :-
        \+ a( _ ),
        !.
faire( [ X | XS ] ) :-
        a( X ),
        \+ ( a( Y ), Y > X ),
        !,
        retract( a( X ) ),
faire( XS ).

f( XS, YS ) :-
        defaire( XS ),
        faire( YS ).
```

que va afficher l'appel suivant :

```
| ?- f( [ 3, 4, 7, 5, 1 ], X ).
```

Réponse :

X = [7,5,4,3,1] ? yes

Numéro 2. (13 pts)
Objectif(s) :

- **Application des connaissances.**
- **Écriture de code Prolog.**

Question : Écrivez une procédure qui calcule l'intersection de deux listes. L'intersection de deux listes est une liste contenant les éléments présents simultanément dans les deux listes. Il n'y a pas d'élément qui se répéte. S'il n'y a pas d'élément dans l'intersection alors la procédure retourne la liste vide. L'ordre des éléments dans la liste résultante n'a pas d'importance. Par exemple :

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

Réponse :

```
inter( [], _, [] ).
inter( [Z | XS], YS, [Z | ZS] ) :-
  member( Z, YS ),
  inter( XS, YS, ZS ).
inter( [Z | XS], YS, ZS ) :-
  \+ member( Z, YS ),
  inter( XS, YS, ZS ).
```

Numéro 3. (13 pts)
Objectif(s) :

- **Application des connaissances.**
- **Lecture de code Prolog.**
- **Utilisation de liste.**
- **Construction de structure.**

Question : Écrivez le code d'une fonction `zipWith`. Cette commande prend en argument un foncteur et deux listes. Elle retourne une liste (en argument).

```
zipWith( Foncteur, Liste1, Liste2, ListeResultat ).
```

par exemple, les appels suivants nous donne :

```
| ?- zipWith( a, [b, c, d], [e,f,g], X ).
X = [a(b,e),a(c,f),a(d,g)] ?
yes
| ?- zipWith( a, [], [], X ).
X = [] ?
yes
| ?- zipWith( F, L1, L2, [b(w, w), b(f, g)] ).
F = b,
L1 = [w,f],
L2 = [w,g] ?
yes
```

Réponse :

```
zipWith( _, [], [], [] ).
zipWith( OP, [A | AS], [B | BS], [CL | RS] ) :-
  functor( CL, OP, 2 ),
  arg( 1, CL, A ),
  arg( 2, CL, B ),
  zipWith( OP, AS, BS, RS ).
```

Numéro 4. (12 pts)
Objectif(s) :

- **Application des connaissances.**
- **Lecture de code Prolog.**

Question : Soit le code suivant :

```
etat(  1, f,  2 ).
etat(  2, o,  3 ).
etat(  2, l,  4 ).
etat(  3, n,  5 ).
etat(  3, r,  6 ).
etat(  4, o,  7 ).
etat(  5, d,  8 ).
etat(  6, t,  9 ).
etat(  6, m, 10 ).
etat(  7, u, 11 ).
etat(  8, u, 12 ).
etat( 10, e, 13 ).

cherche( Y1, [ L | ES ], X ) :- etat( Y1, L, Y2 ),
                                 cherche( Y2, ES, X ).
cherche( Y, [], Y ).
```

a) (8 pts. pts) Que va afficher l'appel suivant :

```
| ?- cherche( 1, Z, 9 ).
```

Réponse :

```
[f,o,r,t].
```

b) (4 pts. pts) Que va afficher l'appel suivant :

```
| ?- cherche( 1, Z, 12 ).
```

Réponse :

```
[f,o,n,d,u].
```