Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

CSI2520 Hiver 2018

Examen de mi-session

Le 27 Février à 16h00

70 minutes

26 points = 26% de votre note finale

Une seule page de notes permise

NOM	**Numéro d'étudiant**

Question	Marks	
1		8
2		6
3		6
4		6
Total		26

<u>Question 1</u> [8 points]

Soit la base de faits Prolog ci-dessous ainsi que le prédicat `trajet`:

```
train(ottawa,toronto).
train(ottawa,montreal).
train(ottawa,sudbury).
train(toronto,kingston).
train(toronto,ottawa).
train(toronto,windsor).
train(montreal,quebec).
train(montreal,kingston).
train(montreal,ste-adele).
train(kingston,cornwall).
trajet([_]).
trajet([C1,C2|L]):-train(C1,C2),trajet([C2|L]).
```

a) Quelles solutions va produire la requête suivante (lister toutes les solutions qui seront trouvées):

```
?- trajet([ottawa,V,kingston,cornwall]).
```

a) Et soit le prédicat suivant :

```
escale(X,Y,Z):-train(X,Y),train(Y,Z),X\=Z.
```

Que sera affiché par les requêtes suivantes?

```
?- setof(X,escale(toronto,ottawa,X),L).
```

```
?- findall(X,escale(toronto,E,X),L).
```

```
?- setof(X,escale(Y,kingston,X),L). ;
```
montrer ici <u>toutes</u> les solutions

Question 2 [6 points] Compléter les prédicats suivants. Vous ne pouvez pas ajouter de nouvelles clauses.

a) Écrire le prédicat `distanceEuclid` calculant la distance Euclidienne entre deux vecteurs. Les vecteurs sont représentés avec une liste.

```
?- distanceEuclid([1,3],[4,7],D).
D=5.0.
```

Cette distance est calculée ainsi $\sqrt{(1-4)^2+(3-7)^2} = 5$. Pour ce faire vous aurez besoin de l'opérateur `sqrt` s'utilisant ainsi: `R is sqrt(25).`

```
distanceEuclid( X, Y, D ) :- distanceEuclid( X, Y, 0, DD),
                             D is sqrt( DD ).

distanceEuclid( [],[],A,A).


distanceEuclid( ____________________,____________________ ,

                ____________________,____________________ ) :-


                ___________________________________________

               distanceEuclid( ____________________,

                               ____________________,

                               ____________________ ,

                               ____________________ ).
```

b) Modifier le prédicat `distanceEuclidM` de façon à fonctionner si les deux listes en entrée ne sont pas de la même longueur. La valeur 0 est assumée pour les éléments manquants de la liste la plus courte.

```
?- distanceEuclidM([1,3,4,2,6],[4,7],D).
D=9.0.
```

Compléter le prédicat ci-dessous. Dans ce cas vous pouvez ajouter de nouvelles règles ou de nouveaux faits.

```
distanceEuclidM( X, Y, D ) :- distanceEuclidM( X, Y, 0, DD),
                              D is sqrt( DD ).

distanceEuclidM( [],[],A,A).


distanceEuclidM( ____________________,____________________ ,

                 ____________________,____________________ ) :-
```

_____

_____

_____

_____

_____

_____

_____

_____

Question 3 [6 points]

Voici un programme Prolog manipulant des arbres binaires:

```
tree(X) :- X =
         t(4,
           t(3,
               nil,
             t(1,
                    t(0, nil, nil),
                    t(2, nil, nil))),
           t(7,
             t(5, nil, nil),
             t(8, nil,
                    t(9, nil, nil)))).

traverse(nil).
traverse(t(X,L,R)) :-
    traverse(L),
    traverse(R),
    write(X),
    write(' ').
```

a) Que sera affiché par la requête ci-dessous? (on veut ici ce que le prédicat `write` va afficher et non la valeur de `X`).

```
?- tree(X), traverse(X).
```

b) Modifier le prédicat `traverse` de façon à obtenir la somme de tous les noeuds.

```
?- tree(X), traverseSum(X,S).
S=39.



traverseSum(_____________________,_____________________ ).


traverseSum(t(X,L,R), S ) :-
```

_____

_____

_____

_____

Question 4 [6 points]

Soit le programme Prolog suivant:

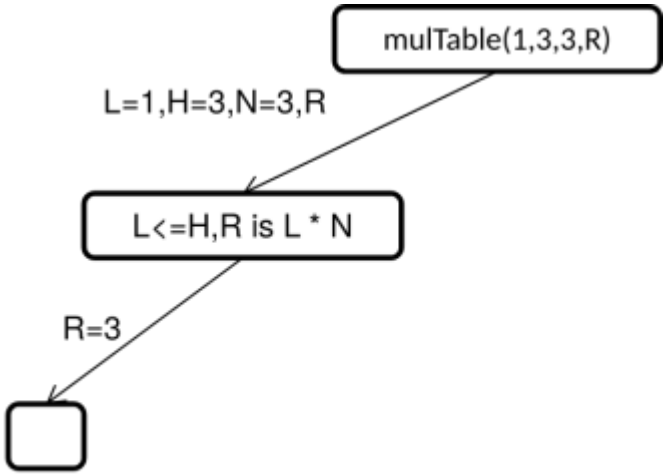
```
mulTable(L,H,N,R) :-
            L =< H,
            R is L * N.

mulTable(L,H,N,R) :-
            L<H,
            LL is L+1,
            mulTable( LL,H,N,R).
```

a) Compléter l'arbre de résolution ci-dessous produit par la requête suivante:

```
?- mulTable(1,3,3,R).
```

Bien montrer toutes les unifications appliquées.



Solution: R=3

b) Écrire une nouvelle version du prédicat `mulTable` qui utilisera une et une seule comparaison (celle montrée ci-dessous). Pour ce faire, ajouter de nouvelles règles utilisant un ou des coupe-choix (*cut* ou *cut-fail*). Vous ne devez pas ajouter de comparaisons (<,>,=:=,etc.).

Exemples:

```
?- mulTable(4,3,3,R).
false.

?- mulTable(1,1,3,R).
3.
```

```
mulTable(L,H,N,R) :- L > H, ; seule comparaison autorisée
```

_____

```
mulTable(H,H,N,R) :-
```

_____

_____

```
mulTable(L,H,N,R) :-
          R is L * N.

mulTable(L,H,N,R) :-
          LL is L+1,
          mulTable( LL,H,N,R).
```