CSI 2520 Midterm Preparation

TUT 1

Exercice I

1. Créer une structure rect qui stocke la hauteur et largeur d'un rectangle

2. Créer une fonction area qui calcule l'aire d'un rectangle

3. Créer une fonction perim qui calcule le périmètre d'un rectangle

4. Créer une fonction main qui utilise les fonctions et la structure ci-dessus pour afficher l'aire et le périmètre d'un rectangle

Exercice II

1. Définir une interface pour les formes géométriques qui contient les méthodes area et périmètre

2. Définir une fonction *measure* qui prend comme paramètre une interface géométrique et affiche l'aire et le périmètre de la forme géométrique.

3. Définir une fonction *main* qui utilise la fonction *measure* pour afficher l'aire et le périmètre d'un carré et d'un cercle.

Exercice V

Compléter le programme suivant afin que la fonction *numberGen* genere des entiers entre *start* et (*start + coun)t,* et la fonction *printNumbers* imprime les entiers générés à l'écran:

```
package main

import "fmt"

func numberGen(start, count int, out chan<- int) {




}

func printNumbers(in <-chan int, done chan<- bool) {




}

func main() {




}
```

Exam exemple

Question 1 [2 points]

Que sera affiché par le programme suivant:

```
affiche([]).
affiche([c|R]):- affiche(R), !, nl.
affiche([X|R]) :- affiche(R), write(X).
```

lorsque la requête suivante est demandée:

```
?- affiche([a,b,c,d,e]).
```

Question 2 [3 points]

Compléter le prédicat ci-dessous comptant le nombre de nombres négatifs dans une liste. Par exemple:

```
?- negatif([0,4,-3,-1,6,-7], N).
N = 3
```

Note: vous ne pouvez pas changer l'ordre des règles présentées.

```
negatif([],0).



negatif([X|L],N) :- _____________________________________________________


negatif([X|L],N) :- X>=0, negatif(L,N).
```

Question 4 [4 points]

Soit les faits suivants listant les personnes possédant un permis de conduire, de pêche ou de port d'arme:

```
permis(robert,peche).
permis(jochen,conduire).
permis(paul,peche).
permis(jean,arme).
permis(jean,conduire).
permis(sam,arme).
permis(sam,peche).
```

a) Donner la requête permettant de trouver une personne n'ayant pas de permis de conduire:

b) Lister dans l'ordre toutes les solutions qui seront trouvées par la requête suivante :

```
?- permis(X,Y),permis(X,Z),Y\==Z.
```

Soit le programme prolog suivant:

```
p(X) :- b(X), c(Y).
p(X) :- a(X).
c(X) :- d(X).
a(1).
a(2).
a(3).
b(4).
b(5).
d(6).
d(7).
```

a) Dessiner l'arbre complet de résolution qui sera généré par la requête suivante (bien montrer l'ordre dans lequel les solutions seront trouvées):

```
?- p(X).
```

Question 7 [6 points]

Soit la base de faits suivante:

```
prerequis(csi2520,csi2510).
prerequis(csi2520,csi2610).
prerequis(csi2510,iti1521).
prerequis(csi2510,mat1748).
prerequis(csi2510,csi2772).
```

Quelle sera la valeur de `L` obtenue par chacune des requêtes suivantes (si plusieurs solutions sont possibles, ne lister que la première qui sera trouvée) :

```
?- bagof(X,Y^prerequis(X,Y),L).
```

```
?- setof(X,Y^prerequis(X,Y),L).
```

```
?- setof(Y,prerequis(X,Y),L).
```

Exercice 1

Soit la base de faits Prolog suivante :

naissance(charles, lisa, philip).
naissance(anne, lisa, philip).
naissance(bob, lisa, philip).
naissance(edward, lisa, philip).
naissance(diana, mary, richard).
naissance(paul, diana, charles).
naissance(john, diana, charles).

Quel sera le résultat de la requête suivante :

?- naissance(S, lisa, Y), naissance(G, M, S)

Donner toutes les solutions qui seront obtenues, dans l'ordre ou elles seront trouvées.

Exercice 2
Soit la base de faits suivantes :
parent(jack,joe). parent(jack,karl). parent(marie,anne). parent(joe,anne). parent(marie,paul). parent(joe,paul). parent(marie,sylvie). parent(bruno,sylvie). parent(anne,zach). parent(tim,zach). parent(sam,tim). parent(emma,tim). parent(josee,sam). parent(gilles,sam). femme(marie). femme(sylvie). femme(anne). femme(emma). femme(josee). homme(karl). homme(jack). homme(joe). homme(bruno). homme(paul). homme(tim). homme(zach). homme(sam). homme(gilles).
Et le prédicat suivant : soeur(X, Y) :- parent(Z, X), parent(Z, Y), femme(X).
Donner, dans l'ordre, toutes les solutions qui seront produites par le prédicat suivant :
?- soeur(X, paul).

Exercice 3

Soit les prédicats suivants créés afin de signifier que deux personnes qui combattent le même ennemi sont des alliés :

```
combat(paul,pierre).
combat(jean,simon).
combat(jean,pierre).
alliés(X,Y) :- combat(X,Z),combat(Y,Z).
```

Lister dans l'ordre toutes les solutions qui seront trouvées par la requête suivante :

```
?- alliés(X,Y).
```

Exercice 6

Considérez la base de faits suivantes:

```
trajet('Windsor', 'Toronto',bus).
trajet('Windsor', 'Detroit',bus).
trajet('Toronto', 'NorthBay',bus).
trajet('Toronto', 'Montreal',bus).
trajet('Toronto', 'Ottawa',bus).
trajet('Toronto', 'Kingston',bateau).
trajet('Kingston', 'Ottawa',bateau).
trajet('Montreal', 'Ottawa',bus).
```

Donnez les requêtes suivantes:

1. y a t il un chemin entre Windsor et Ottawa?

2. Peut-on visiter la ville du Québec?

3. Imprimer les villes qui sont dans le trajet de Toronto à Ottawa.

4. Peut on avoir un chemin agréable entre Windsor et Kingston? (un chemin est dit agréable lorsqu'il alterne un trajet en bus et un trajet en bateau!).

Considérez la base de faits suivantes:

```
p(a,b).
p(a,c).
p(c,d).
p(d,e).
p(d,f).
p(n,w).

q(A,B) :- p(A,B).
q(A,B) :- p(X,B), q(A,X).
```

Dessiner l'arbre de resolution pour la requète suivante :

```
?- q(a,e)
```

Exercice 1

1. Ecrire un prédicat qui convertit une température en degrés Celsius à son équivalent en température en degrés Fahrenheit. La formule de conversion est donnée par l'expression suivante :

$F = C * 9/5 + 32$

2. Ecrire un prédicat qui détermine si une température (ºF) est au dessous du point de congélation (32ºF).

Exercice 2

L'heure du jour est exprimée en heures, minutes et secondes, comme *c(H, M, S)*. On veut écrie un prédicat qui va produire exactement une seule solution : l'heure après une seconde. Voici quelques exemples:

Exercice 1

Soit la base de faits et prédicats suivants :

```
p(a).                          /* #1 */
p(X) :- q(X), r(X).            /* #2 */
p(X) :- u(X).                  /* #3 */
q(X) :- s(X).                  /* #4 */
r(a).                          /* #5 */
r(b).                          /* #6 */
s(a).                          /* #7 */
s(b).                          /* #8 */
s(c).                          /* #9 */
u(d).                          /* #10 */
```

1. Tracer l'arbre de recherche de P(X)

Exercice 3

Soit la base de faits et prédicats suivants :

```
s(X,Y):- q(X,Y).
s(0,0).
q(X,Y):- i(X), j(Y).
i(1).
i(2).
j(1).
j(2).
j(3).
```

1. Déterminer le résultat de la requête suivante en dessinant l'arbre de recherche.

Exercice 4
Soit la base de faits suivante:
p(a).
p(b).
q(c).
1. Quel est le résultat des requêtes suivantes:

?-not p(X), q(X).

?-q(X), not p(X).

Soit la base de faits et prédicats suivants :
sad(X) :- \+ happy(X).
happy(X) :- beautiful(X), rich(X).
rich(bill).
beautiful(michael).
rich(michael).
beautiful(cinderella).
1. Quel est le résultat des requêtes suivantes:
?- sad(bill).

?- sad(cinderella).

?- sad(michael).

?- sad(jim).

?- sad(Someone).

Question 1 Règles Prolog [4 points]

Voici une règle définissant l'éligibilité d'un étudiant au programme Coop:

1) L'étudiant(e) doit être inscrit(e) en 2ième année ou plus,
2) L'étudiant(e) doit avoir passé le cours ITI1121 avec une note d'au moins D+,
3) L'étudiant(e) doit avoir passé le cours CSI2120 avec une note d'au moins B, et
4) L'étudiant(e) doit être inscrit(e) à temps plein.

Concevoir une règle Prolog vérifiant les conditions énumérées, considérant une base de faits dans le format suivant :

```
% Nom, Année, status (temps plein/ temps partiel), [ (Cours, Note), ...]
student( jane, 2, ft, [ (iti1120,'B'), (iti1121,'B+'),
(csi2120,'A+'), (csi2372,'A-') ]).
student( joe, 3, pt, [ (iti1120,'A'), (iti1121,'B'), (csi2120,'C'),
(csi2372,'C'), (csi3105,'F')]).
student( mary, 1, ft, [ (iti1100, 'A+'), (iti1120,'A+'),
(iti1121,'A+') ] ).

% vrai si le cours se trouve dans la liste L avec une note d'au moins MinGrade
grade(Course, MinGrade, [(Course,Grade)|_] ) :- Grade @=< MinGrade.
grade(Course, MinGrade, [_|L]) :- grade(Course, MinGrade, L).

eligible( Name ) :-
```

Question 2 Liste Prolog [3 points]

Le prédicat `deleteBack` détruit la dernière occurrence de l'élément `R` dans la liste `L`. Compléter ce prédicat.

```
deleteBack(_____________________________,_____________________________ ,

          _____________________________ , _____________________________  ).

deleteBack( R, [R|LI], [R|A], L0) :-
          \+member(R,LI), !,
          deleteBack( R, LI, A, L0 ).

deleteBack(_____________________________,_____________________________,

          _____________________________ , _____________________________ ):-

              deleteBack(_____________________________,

                         _____________________________,

                         _____________________________,

                         _____________________________).
```