EISTI - DÉPARTEMENT INFORMATIQUE

EXAMEN DE LOGIQUE COMPUTATIONNELLE ET PROLOG

28 juin 2012 – **DURÉE 3h00**

La consultation et l'échange des documents, et l'utilisation des calculatrices sont interdits.

L'utilisation des 3 feuilles manuscrites recto-verso, format A4 est autorisée

NOM : .....

NOTE

DÉTAIL

		1a	**1b**	**1c**	**1d**	**2**					
Exercice 1.											
		1	**2**	**3**	**4**						
Exercice 2											
Exercice 3.											
		1	**2**	**3**							
Exercice 4.											
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	
Exercice 5.											
		1	**2**	**3**							
Exercice 6.											
		1	**2**	**3**							
Exercice 7.											

Le corrigé se trouve sur `http://sifoci.eisti.fr`, rubrique Logique computationnelle.

NOM : .....

Exercice 1.- Logique propositionnelle

On suppose que l'on a les règles et faits suivants :

(a) Si Toto rate son examen, alors Toto sera déprimé.

(b) Si Toto ne va pas à la piscine, il sera déprimé.

(c) À la piscine, Toto ne travaille pas.

(d) Toto ratera son examen s'il ne travaille pas.

1. Modéliser l'énoncé à l'aide de formules de la logique propositionnelle.

NOM : ....

2. Prouver que Toto sera toujours déprimé.

NOM : .....

Exercice 2.- Logique des prédicats

Une école (disons l'EISTI) comporte m classes, chacune contenant un certain nombre d'élèves. Après les soutenances le BDE souhaite organiser un repas qui réunisse tous les élèves autour de n tables.

Les énoncés suivants sont supposés vrais :

(a) Tout élève de l'école appartient à une seule classe.

(b) Pour apprécier une personne il faut la connaître.

(c) Les élèves d'une même classe se connaissent et s'apprécient.

(d) Toute personne ne mange que sur une seule table.

(e) Dans un repas réussi, chacun apprécie les élèves à sa table ou alors il ne les connaît pas.

On souhaite décrire cette situation en n'utilisant que des prédicats portant sur les élèves, et on interprétera les formules obtenues dans un domaine qui ne contient que des élèves. On utilisera le langage suivant :

- Des prédicats unaires $\texttt{classe}_1, \texttt{classe}_2, \cdots$: par exemple $\texttt{classe}_1\texttt{(toto)}$ signifie que `toto` est dans la $\texttt{classe}_1$.
- Des prédicats binaires $\texttt{table}_1, \texttt{table}_2, \cdots$: par exemple $\texttt{table}_2\texttt{(titi)}$ signifie que `titi` mange à la $\texttt{table}_2$.
- Un prédicat binaire `connait` : par exemple `connait(toto,titi)` signifie que `toto` connaît `titi`.
- Un prédicat binaire `apprecie` : par exemple `apprecie(toto,titi)` signifie que `toto` apprécie `titi`.

On suppose qu'il y a trois classes et que le repas comporte deux tables.

NOM : .....

1. Écrire dans le calcul des prédicats les énoncés (a) à (e) avec le langage de description donné.

NOM : .....

2. Donner un modèle qui satisfasse tous les énoncés sauf le dernier.

NOM : ....

3. Donner un modèle qui satisfasse tous les énoncés.

NOM : .....

4. Avec le langage proposé, pourquoi serait-il impossible d'exprimer par une formule l'énoncé 1 si on ne connaissait pas le nombre de classes ?

NOM : .....

Exercice 3.- Forme normale conjonctive
Soit la fbf

$$\forall X\left(p\left(X\right)\rightarrow\left(\exists Y\ q\left(X,Y\right)\vee r\left(Y\right)\right)\right)\rightarrow\exists X\left(w\left(X\right)\rightarrow\left(\forall Y q\left(Y,X\right)\vee r\left(X\right)\right)\right)$$

Calculer sa forme normale conjonctive.

NOM : .....

Exercice 4.- Modèles. Univers et base de Herbrand
Soit le programme défini :
`P = {ami(toto,koko).}`

1. Donner l'univers de Herbrand

2. Donner la base de Herbrand.

NOM : .....

3. Considérons un langage $\mathcal{L}$ muni d'un prédicat $p/2$ d'arité 2. Trouver une interprétation de Herbrand $\mathcal{I}$, issue de la base de Herbrand, qui est un modèle de $\exists X \; \exists Y \; p\,(X, Y)$.

NOM : .....

Exercice 5.- Propriétés des opérateurs Prolog
Donner en les justifiant précisément les réponses de Prolog aux questions suivantes.

1. `?- X=2.`

2. `?- 2=X.`

3. `?- X =:= 2.`

NOM : ....

4. `?- 2+2 =:= 4.`

5. `?- X is 2.`

6. `?- 2 is X.`

NOM : ....

7. `?- 2+2 is 4.`

8. `?- 4 is 2+2.`

NOM : ....

9. `?- \+ member(X,[]).`

10. `?- \+ member(X,[1,2,3]).`

NOM : .....

Exercice 6.- Listes

On se place dans le cadre des nombres entiers positifs ou nuls.

1. Écrire le prédicat `pair(N)` qui est vrai si `N` est un nombre pair.

NOM : .....

2. Écrire le prédicat `impair(N)` qui est vrai si `N` est un nombre impair.

NOM : .....

3. Écrire le prédicat `pairsimpairs(L,P,I)` où `L` est une liste de nombres entiers en entrée, et les listes `P`, `I` sont deux sous-listes en sortie contenant les nombres pairs et les nombres impairs respectivement de la liste `L`.

NOM : .....

Exercice 7.- Mot le plus long

Le jeu « le mot le plus long » a été popularisé par l'émission « les chiffres et les lettres ». Le principe de ce jeu est le suivant : on tire des lettres puis on essaye de former le mot le plus long possible avec ces des lettres.

Vous devez écrire un programme Prolog qui permet de jouer à ce jeu. Pour ce faire, vous disposez des prédicat suivants :

- `dico(Dico)` avec `Dico`, la liste des mots existants sous la forme `[[b,o,n],[b,o,n,j,o,u,r],[p,o,n,t],[p,o,u,r],[j,o,u,r],...]`
- `lettres(Lettres)` avec `Lettres`, un tirage aléatoire de lettres sous la forme `[n,j,r,j,o,u,m,t,p]`.
- `ensembleMots(Dico,Lettres,4,Mots)` avec `Mots`, l'ensemble des mots de longueur `Taille` formés à partir de `Lettres` et appartenant à la liste `Dico`
- `estUnMot(Dico,Perm)` qui est vrai si `Perm` est un mot existant dans la liste `Dico`.
- `perm(Lettres,Taille,Perm)` avec `Perm` une permutation de `Taille` lettres prises parmi la liste `Lettres`.

Voici un programme de test :

```
dico(Dico),
lettres(Lettres),
unMotLePlusLong(Dico,Lettres,Mot),
unAutreMotLePlusLong(Dico,Lettres,Mot,Autre),
ensembleMots(Dico,Lettres,4,Mots),
tousLesMots(Dico,Lettres,TousMots),nl,
write('lettres de depart : '),write(Lettres),nl,
write('un mot le plus long : '),write(Mot),nl,
write('un autre mot le plus long : '),write(Autre),nl,
write('ensemble des mots de taille 4 : '),write(Mots),nl,
write('tous les mots possibles : '),write(TousMots),nl,nl.
```

L'exécution de ce programme donne le résultat suivant :

```
lettres de depart : [n,x,r,j,y,u,w,b,p]
un mot le plus long : [b,r,u,n]
un autre mot le plus long : [j,u,r,y]
ensemble des mots de taille 4 : [[b,r,u,n],[j,u,r,y]]
tous les mots possibles : [[b,r,u],[b,r,u,n],[b,u],
                           [j,u,r,y],[n,u],[p,u],[p,u,b],
                           [p,u,r],[p,u,y],[u,n]]
```

NOM : .....

Vous devez écrire les prédicats suivants :

1. `unMotLePlusLong(Dico,Lettres,Mot)`
   ⇒ `Mot` est un mot le plus long possible, formé à partir de `Lettres` et appartenant à la liste `Dico`.

NOM : .....

2. `unAutreMotLePlusLong(Dico,Lettres,Mot,Autre)`
   ⇒ `Autre` est un mot le plus long possible, différent de `Mot`, formé à partir de `Lettres` et appartenant à la liste `Dico`.

NOM : .....

3. `tousLesMotsPossibles(Dico,Lettres,Mots)`
   ⇒ `Mots` est l'ensemble de tous les mots de `Dico` qu'il est possible de former à partir des lettres contenues dans `Lettres`.