

CORRIGÉ INTERRO DE PROLOG

1. À l'aide de la méthode des tables de vérité, dites, pour l'interprétation selon laquelle p est fausse, q est vraie et r est fausse, quelle est la valeur de vérité de la proposition suivante : (2 pts)

$$r \Rightarrow ((p \vee q) \wedge \neg(p \wedge q))$$

2. Traduisez les propositions suivantes de la langue naturelle en langage du calcul des prédicats, et pour chacune de ces traductions, trouvez sa négation, puis proposez une phrase en langue naturelle qui convient : (6 pts)

- a. Zed a triché à l'examen, mais il ne triche pas aux examens.

$$\star \text{ Triche}(\text{zed}, \text{examen}) \wedge \neg \text{Triche}(\text{zed}, \text{examen})$$

$$\star \text{ négation : } \neg \text{Triche}(\text{zed}, \text{examen}) \vee \text{Triche}(\text{zed}, \text{examen})$$

$$\star \text{ Zed n'a pas triché à l'examen, ou alors il triche aux examens.}$$

- b. Si les pays africains sont pauvres, alors les africains ne sont pas libres.

$$\star \forall x(\text{PaysAfricain}(x) \Rightarrow \text{Pauvre}(x)) \Rightarrow \forall y(\text{Africain}(y) \Rightarrow \neg \text{Libre}(y))$$

$$\star \text{ négation : } \forall x(\text{PaysAfricain}(x) \Rightarrow \text{Pauvre}(x)) \wedge \exists y(\text{Africain} \wedge \text{Libre}(y))$$

$$\star \text{ Les pays africains sont pauvres, mais il y a des africains qui sont libres.}$$

3. Le prédicat croissant(N) est censé afficher les nombres entiers de 1 à N. Voici sa définition :

```
croissant(0).
croissant(N):- N>0,
                writeln(N),
                N1 is N-1,
                croissant(N1).
```

Pourtant, lorsqu'on lance la requête ?- croissant(5). le résultat apparaît comme suit :

```
5
4
3
2
1
```

Proposez ce qu'il faudrait faire pour obtenir le résultat attendu. (2 pts)

```
croissant(0).
```

```
croissant(N):- N>0,
               N1 is N-1,
               croissant(N1),
               writeln(N).
```

4. Écrire le prédicat `prodList(L1,L2,L)` qui met dans la liste **L** le produit des éléments correspondants des listes **L1** et **L2**. Lorsque l'une des deux listes est vide, ce prédicat ajoute à **L** les éléments restants de la liste non vide. (4 pts)

```
prodList([],L,L).
prodList(L,[],L).
prodList([X|L1],[Y|L2],[Z|L]):- Z is X*Y, prodList(L1,L2,L).
```

5. Écrire le prédicat `minListe(L,R1,R2)` dont **R1** et **R2** sont les deux plus petits éléments distincts d'une liste **L** de nombres. (6 pts)

```
min(X,Y,X):-X=<Y.
min(X,Y,Y):- Y<X.

elementMin([X],X).
elementMin([T|Q],Min):- elementMin(Q,Min1),min(T,Min1,Min).

minListe(L,R1,R2):- elementMin(L,R1),
                    delete(L,R1,LL),
                    elementMin(LL,R2).
```