

Compte Rendu TP1

Question1

```
voiture(rouge).
voiture(vert(clair)).
voiture(gris).
voiture(blanc).

bateau(vert(_)).
bateau(noir).
bateau(blanc).

choixCouleur(CouleurBateau,CouleurVoiture):-
    bateau(CouleurBateau),
    voiture(CouleurVoiture).
```

Tests

```
choixCouleur(noir,vert(clair)).

Yes (0.00s cpu)
```

Question2

Prolog est un solveur de contrainte sur le domaine des arbres car les contraintes sur les arbres sont des contraintes passives, et prolog sait résoudre les contraintes passives

Question3

```
isBetween(Min,Min,Max):-
    Max >= Min.
isBetween(Var,Min,Max):-
    X is Min+1,
    X=<Max,
    isBetween(Var,X,Max).
```

Tests

```
?- isBetween(5,2,7).
Yes (0.00s cpu, solution 1, maybe more)

?- isBetween(X,6,8).
[eclipse 4]: ?- isBetween(X,6,8).

X = 6
Yes (0.00s cpu, solution 1, maybe more) ? ;

X = 7
Yes (0.00s cpu, solution 2, maybe more) ? ;
```

X = 8
Yes (0.00s cpu, solution 3, maybe more) ? ;

No (0.00s cpu)

Question4

```
commande(NbResistance,NbCondensateur):-  
    isBetween(NbResistance,5000,10000),  
    isBetween(NbCondensateur,9000,20000),  
    NbResistance>=NbCondensateur.
```

Tests

```
commande(X,Y).  
[eclipse 5]: commande(X,Y).  
lists.eco  loaded in 0.00 seconds
```

X = 9001
Y = 9000
Yes (0.00s cpu, solution 1, maybe more) ? ;

X = 9002
Y = 9000
Yes (0.00s cpu, solution 2, maybe more) ? ;

X = 9002
Y = 9001
Yes (0.00s cpu, solution 3, maybe more) ? ;

X = 9003
Y = 9000

```
commande(6500,9500). No  
commande(10000,8000). No
```

Question5

commande (N1, N2)

isBetween (N1, 5000, 10000),
isBetween (N2, 9000, 20000),
N1 >= N2.

Min = 5000, Max = 10000

N1 = 5000

isBetween (5000, 5000, 10000),
isBetween (N2, 9000, 20000),
N1 >= N2.

N2 = 9000

isBetween (5000, 5000, 10000)
isBetween (9000, 9000, 20000)
5000 >= 9000.

fail

N2 = 9001

isBetween (5000, 5000, 10000)
isBetween (9001, 9000, 20000)
5000 >= 9001.

fail

N2 = 9002

isBetween (5000, 5000, 10000),
isBetween (9002, 9000, 20000),
5000 >= 9002.

fail

Question6

On ne peut pas comparer les valeurs avant de les avoir instanciées,
donc on doit les générer avant de pouvoir les tester

Question7

```
commande(NbResistance, NbCondensateur):-  
    NbResistance#::(5000..10000),  
    NbCondensateur#::(9000..20000),  
    NbResistance#>NbCondensateur.
```

Tests


```
commande(X,Y).
X = X{9001 .. 10000}
Y = Y{9000 .. 9999}
Ce résultat donne les nouveau domaines possibles mais n'instancie pas les variables
```

Question8

```
commande(NbResistance,NbCondensateur):-
    NbResistance#::(5000..10000),
    NbCondensateur#::(9000..20000),
    NbResistance#>NbCondensateur,
    labeling([NbResistance,NbCondensateur]).
```

Tests

```
[eclipse 10]: commande(X,Y).

X = 9001
Y = 9000
Yes (0.00s cpu, solution 1, maybe more) ? ;

X = 9002
Y = 9000
Yes (0.01s cpu, solution 2, maybe more) ? ;

X = 9002
Y = 9001
Yes (0.01s cpu, solution 3, maybe more) ? ;

X = 9003
Y = 9000
Yes (0.01s cpu, solution 4, maybe more) ?
```

Question9

```
chapie(Chats,Pies,Pattes,Tetes):-
    Chats#>=0,
    Pies#>=0,
    Pattes#>=0,
    Tetes#>=0,
    Pattes#=Chats*4+Pies*2,
    Tetes#=Chats+Pies.
```

Tests

```
chapie(2,X,Y,5).
X = 3
Y = 14
Yes (0.00s cpu)
```

Question10

chapie(Chats,Pies,X,Y), X#=Y*3,Pies#<1000,labeling([Chats,Pies,X,Y]).

Tests

```
Chats = 0
Pies = 0
X = 0
Y = 0
Yes (0.00s cpu, solution 1, maybe more) ? ;

Chats = 1
Pies = 1
X = 6
Y = 2
Yes (0.01s cpu, solution 2, maybe more) ?
...
```

Question11

```
/**or/2*/
vabs(Val,AbsVal):-
    (Val#>=0,Val#=AbsVal) or (Val#<0,AbsVal #= 0-Val).
```

Tests

```
[eclipse 91]: ?- vabs(-5,Y).

Y = 5
Yes (0.00s cpu)
[eclipse 92]: ?- vabs(4,X).

X = 4
Yes (0.00s cpu)
[eclipse 93]: ?- vabs(-8,8).

Yes (0.00s cpu)
```

```
/**Point de choix

vabs(Val,Val):-
    Val#>=0.
vabs(Val,AbsVal):-
    Val#<0,
    AbsVal #= 0-Val.
```

Tests

```
?- vabs(4,X).

X = 4
Yes (0.00s cpu, solution 1, maybe more) ?
?- vabs(-5,Y).

Y = 5
Yes (0.00s cpu)
?- vabs(-8,8).

Yes (0.00s cpu)
```

Question12

```
/**or/2
X#:: -10..10,vabs(X,Y),labeling([X,Y]).
```

Tests

```
Echantillon :
X = -4
Y = 4
Yes (0.00s cpu, solution 7, maybe more) ? ;

X = -3
Y = 3
Yes (0.00s cpu, solution 8, maybe more) ? ;

X = -2
Y = 2
Yes (0.00s cpu, solution 9, maybe more) ? ;

X = -1
Y = 1
Yes (0.00s cpu, solution 10, maybe more) ? ;

X = 0
Y = 0
Yes (0.00s cpu, solution 11, maybe more) ? ;

X = 1
Y = 1
Yes (0.00s cpu, solution 12, maybe more) ? ;

X = 2
Y = 2
Yes (0.00s cpu, solution 13, maybe more) ? ;

X = 3
Y = 3
Yes (0.00s cpu, solution 14, maybe more) ? ;

X = 4
Y = 4
Yes (0.00s cpu, solution 15, maybe more) ? ;

X = 5
Y = 5
Yes (0.00s cpu, solution 16, maybe more) ? ;
```

```

X = 6
Y = 6
Yes (0.00s cpu, solution 17, maybe more) ? ;

X = 7
Y = 7
Yes (0.01s cpu, solution 18, maybe more) ? ;

X = 8
Y = 8
Yes (0.01s cpu, solution 19, maybe more) ? ;
*/

/**Point de choix
X#:: -10..10,vabs(X,Y),labeling([X,Y]).

```

Tests

```

Echantillon
X = 5
Y = 5
Yes (0.00s cpu, solution 6, maybe more) ? ;

X = 6
Y = 6
Yes (0.00s cpu, solution 7, maybe more) ? ;

X = 7
Y = 7
Yes (0.00s cpu, solution 8, maybe more) ? ;

X = 8
Y = 8
Yes (0.00s cpu, solution 9, maybe more) ? ;

X = 9
Y = 9
Yes (0.00s cpu, solution 10, maybe more) ? ;

X = 10
Y = 10
Yes (0.00s cpu, solution 11, maybe more) ? ;

X = -10
Y = 10
Yes (0.00s cpu, solution 12, maybe more) ? ;

X = -9
Y = 9
Yes (0.00s cpu, solution 13, maybe more) ? ;

X = -8
Y = 8
Yes (0.00s cpu, solution 14, maybe more) ? ;

X = -7
Y = 7
Yes (0.00s cpu, solution 15, maybe more) ? ;

X = -6
Y = 6
Yes (0.00s cpu, solution 16, maybe more) ?

```

La différence principale est l'ordre dans lequel les variables sont instanciées
Avec le or, tous l'intervalle est testé en même temps,
avec le point de choix il est d'abord séparé e deux

Question13

```
faitListe(ListVar,Taille,Min,Max):-
    length(ListVar,Taille),
    ( foreach(Elem,ListVar),
      param(Min,Max)
    do
        Elem#::Min..Max
    ).
```

Tests

//

Question14

```
faitSuite([_,_]).
faitSuite([X,Y,Z|R]):-
    vabs(Y,Yabs),
    Z #= Yabs - X,
    faitSuite([Y,Z|R]).
```

Tests

[eclipse 10]: faitSuite([1,2,3,4,5]).

Yes (0.00s cpu)

Question15

```
periode_neuf([X,Y,_,_,_,_,_,_,_,X1,Y1]):-
    X#=X1,
    Y#=Y1.

contre_exemple:-
    X#::(-100..100),
    Y#::(-100..100),
    faitSuite([X,Y,_,_,_,_,_,_,_,X1,Y1]),
    (X#\=X1) or (Y#\=Y1),
    labeling([X,Y]).
```


periode_neuf([1,2,3,4,5,6,7,8,9]).

Yes (0.00s cpu)

Pour avoir une période de 9 il suffit que le 10 élément soit égal au 1er et le 11e au 2e, puisque la suite est définie par ses deux derniers éléments lorsqu'on execute contre_exemple, on obtient No, ce qui signifie qu'aucune liste qui satisfait la suite n'est pas de période 9 -> Toutes les listes satisfaisant la suite sont de période 9