LANFRANCHI Oriane - P24

TP : Mastermind en Prolog

L'ensemble du programme se trouve dans le document *Mastermind.txt*. Il est possible de copier-coller l'entièreté du document *.txt* pour tester le code (sur Swish par exemple). Ce document-ci est une mise au propre du TP.

Pour lancer le programme (et jouer au mastermind), il faut faire la requête jouons(M, N, Max), par exemple : jouons(6, 4, 10).

Question 1

1. nBienPlace(+Code1, +Code2, -BP)

```
nBienPlace([], [], 0).

nBienPlace([H|T1], [H|T2], BP) :-
    nBienPlace(T1, T2, BP1),
    BP is BP1+1.

nBienPlace([H|T1], [H1|T2], BP) :-
    H\=H1,
    nBienPlace(T1, T2, BP1),
    BP is BP1.
```

2. gagne(+Code1, +Code2)

```
longueur([], 0).

longueur([_|R], Lg) :-
    longueur(R, LgTmp),
    Lg is LgTmp+1.

gagne(Code1, Code2) :-
    longueur(Code1, L),
    longueur(Code2, L),
    nBienPlace(Code1, Code2, L).
```

Question 2

1. element(+E, +L)

```
element(E, [E|_]).
element(E, [_|T]) :- element(E, T).
```

2. enleve(+E, +L1, -L2)

```
enleve(_, [], []).
```

```
enleve(E, [E|T], T).

enleve(E, [X|T], [X|Y]) :-
    E\=X,
    enleve(E, T, Y).
```

3. enleveBP(+Code1, +Code2, -Code1Bis, -Code2Bis)

Pour cette question, il y a besoin d'utiliser la concaténation, définie telle que suit :

```
concat([], L, L).
concat([H|T], L2, [H|LTmp]) :- concat(T, L2, LTmp).
```

Note : En ce qui concerne le morceau de code suivant : il ne répond pas à la question posée dans l'énoncé du TP. J'ai, dans un premier temps, mal lu la question. Je compris que Code1Bis devait contenir tous les éléments de Code1 non présents dans Code2 (et inversément pour Code2Bis).

Le code ci-après n'est donc pas adéquat, mais j'avais mal au coeur de devoir le supprimer après avoir passé un peu de temps dessus. Le voici donc, même s'il peut être ignoré (la bonne version de ma réponse se trouve juste après).

```
enleveBPCode(_, [], []).

enleveBPCode(Code1, [H|T], Code2Bis) :-
    element(H, Code1),
    enleveBPCode(Code1, T, Code2Bis).

enleveBPCode(Code1, [H|T], Code2Bis) :-
    \+element(H, Code1),
    enleveBPCode(Code1, T, Code2BisTmp),
    concat([H], Code2BisTmp, Code2Bis).

enleveBP(Code, Code, [], []).

enleveBP(Code1, Code2, Code1Bis, Code2Bis) :-
    enleveBPCode(Code1, Code2, Code2Bis),
    enleveBPCode(Code2, Code1, Code1Bis).
```

La réponse à la question (la bonne, cette fois) :

```
enleveBP([], [], [], []).

enleveBP([H1|T1], [H2|T2], Code1Bis, Code2Bis) :-
    H1\=H2,
    enleveBP(T1, T2, Code1BisTmp, Code2BisTmp),
    concat([H1], Code1BisTmp, Code1Bis),
    concat([H2], Code2BisTmp, Code2Bis).

enleveBP([H|T1], [H|T2], Code1Bis, Code2Bis) :-
    enleveBP(T1, T2, Code1BisTmp, Code2BisTmp),
```

```
    concat([], Code1BisTmp, Code1Bis),
    concat([], Code2BisTmp, Code2Bis).
```

4.nMalPlaces(+Code1, +Code2, -MP)

```
nMalPlacesAux([], _, 0).

nMalPlacesAux([H1|T1], L, MP) :-
    element(H1, L),
    enleve(H1, L, L2),
    nMalPlacesAux(T1, L2, MP2),
    MP is MP2 + 1.

nMalPlacesAux([H1|T1], L, MP) :-
    \+element(H1, L),
    nMalPlacesAux(T1, L, MP2),
    MP is MP2.

nMalPlaces(Code, Code, 0).

nMalPlaces(Code1, Code2, MP) :-
    enleveBP(Code1, Code2, Code1Bis, Code2Bis),
    nMalPlacesAux(Code1Bis, Code2Bis, MP2),
    MP is MP2.
```

QUESTION 3

1. codeur(+M, +N, -Code)

```
codeur(_, 0, []).

codeur(M, N, Code) :-
    N > 0,
    NTmp is N-1,
    codeur(M, NTmp, CodeTmp),
    random(1, M, Couleur),
    concat([Couleur], CodeTmp, Code).
```

QUESTION 4

jouons(+M, +N, +Max)

```
finDeTour(Code, Tentative, 0) :-
    Code\=Tentative,
    write("Perdu !").

finDeTour(Code, Code, _) :- write("Gagné !").

finDeTour(Code, Tentative, CoupsRestants) :-
    Code\=Tentative,
```

```
    CoupsRestants > 0,
    gameplay(Code, CoupsRestants).

gameplay(Code, CoupsRestants) :-
    write("Il reste "),
    write(CoupsRestants),
    write(" coup(s)."), nl,
    write("Donner un code : "),
    read(Tentative),
    nBienPlace(Code, Tentative, BP),
    nMalPlaces(Code, Tentative, MP),
    write("BP : "),
    write(BP),
    write("/MP :"),
    write(MP), nl,
    CoupsRestantsProchain is CoupsRestants-1,
    finDeTour(Code, Tentative, CoupsRestantsProchain).

jouons(M, N, Max) :-
    codeur(M, N, Code),
    gameplay(Code, Max).
```