Compte-rendu TD10

Prédicats retourneUtile/3 et retourne/2 pour renverser une liste qu'on utilisera pour renverser les listes de cases, plus précisément les chemins solutions du labyrinthe

```
retourneUtile([T|Q],X,L) :- retourneUtile(Q,X,[T|L]).
retourneUtile([],L,L).
retourne(L,R) :- retourneUtile(L,R,[]).
```

Prédicat dessine_chemin/2 pour colorier les cases d'une liste de case

```
dessine_chemin([], _).
dessine_chemin([ case(X,Y) | Q ], C) :-
    laby(X,Y, V), V \== 3,
    N is 19*Y+X,
    gr_rect_couleur(N, C),
    dessine_chemin(Q, C)
.
dessine_chemin([ case(X,Y) | Q], C) :-
laby(X,Y, 3),
dessine_chemin(Q,C).
```

Prédicat avanceDe/2 qui cherche les solutions d'un labyrinthe et dessine les résultats

```
avanceDe(X,Y,_) :- sleep(0.01), rougeXY(X,Y),fail.

avanceDe( X, Y, L) :-laby(X,Y,3),
      retourne([case(X,Y)|L],Chemin),
      dessine_chemin(Chemin,yellow),
      sleep(0.5),fail.

avanceDe(X,Y, ListeAriane) :-
    laby(X,Y,R),
    R \== 1,
    Y1 is Y - 1,
    laby(X,Y1,R1),
    R1 \== 1,
    \+ member(case(X,Y1),ListeAriane),
    avanceDe(X,Y1,[case(X,Y) |ListeAriane]).


avanceDe(X,Y, ListeAriane) :-
    laby(X,Y,R),
    R \== 1,
    X1 is X - 1,
    laby(X1,Y,R1),
    R1 \==1,
    \+ member(case(X1,Y),ListeAriane),
    avanceDe(X1,Y, [case(X,Y) |ListeAriane]).

avanceDe(X,Y, ListeAriane) :-
    laby(X,Y,R),
    R \== 1,
    X1 is X + 1,
    laby(X1,Y,R1),
    R1 \== 1,
```

```
    \+ member(case(X1,Y),ListeAriane),
    avanceDe(X1,Y, [case(X,Y) |ListeAriane]).

avanceDe(X,Y, ListeAriane) :-
    laby(X,Y,R),
    R \== 1,
    Y1 is Y + 1,
    laby(X,Y1,R1),
    R1 \== 1,
    \+ member(case(X,Y1),ListeAriane),
    avanceDe(X,Y1, [case(X,Y) |ListeAriane]).

avanceDe(X,Y,_) :- blancXY(X,Y), fail.
```

Prédicat avanceDeCollect/4 qui génère les chemins solutions C du labyrinthe

```
avanceDeCollect( X, Y, L,Chemin) :-laby(X,Y, 3),
     retourne([case(X,Y)|L],Chemin).

avanceDeCollect(X,Y, ListeAriane,C) :-
    laby(X,Y,R),
    R \== 1,
    Y1 is Y - 1,
    laby(X,Y1,R1),
    R1 \== 1,
    \+ member(case(X,Y1),ListeAriane),
    avanceDeCollect(X,Y1,[case(X,Y) |ListeAriane],C).


avanceDeCollect(X,Y, ListeAriane,C) :-
    laby(X,Y,R),
    R \== 1,
    X1 is X - 1,
    laby(X1,Y,R1),
    R1 \==1,
    \+ member(case(X1,Y),ListeAriane),
    avanceDeCollect(X1,Y, [case(X,Y) |ListeAriane],C).

avanceDeCollect(X,Y, ListeAriane,C) :-
    laby(X,Y,R),
    R \== 1,
    X1 is X + 1,
    laby(X1,Y,R1),
    R1 \== 1,
    \+ member(case(X1,Y),ListeAriane),
    avanceDeCollect(X1,Y, [case(X,Y) |ListeAriane],C).

avanceDeCollect(X,Y, ListeAriane,C) :-
    laby(X,Y,R),
    R \== 1,
    Y1 is Y + 1,
    laby(X,Y1,R1),
    R1 \== 1,
    \+ member(case(X,Y1),ListeAriane),
    avanceDeCollect(X,Y1, [case(X,Y) |ListeAriane],C).
```

```
?- gr_init,dessine_laby,avanceDeCollect(1,1,[],C).
C = [case(1, 1), case(1, 2), case(1, 3), case(1, 4), case(1, 5), case(2, 5), case(3, 5), case(3, 4), case(..., ...)|...] ;
C = [case(1, 1), case(1, 2), case(1, 3), case(1, 4), case(1, 5), case(2, 5), case(3, 5), case(3, 4), case(..., ...)|...] ;
C = [case(1, 1), case(1, 2), case(1, 3), case(1, 4), case(1, 5), case(2, 5), case(3, 5), case(3, 4), case(..., ...)|...] ;
C = [case(1, 1), case(1, 2), case(1, 3), case(1, 4), case(1, 5), case(2, 5), case(3, 5), case(3, 4), case(..., ...)|...] ;
C = [case(1, 1), case(1, 2), case(1, 3), case(1, 4), case(1, 5), case(2, 5), case(3, 5), case(3, 6), case(..., ...)|...] ;
C = [case(1, 1), case(1, 2), case(1, 3), case(1, 4), case(1, 5), case(2, 5), case(3, 5), case(3, 6), case(..., ...)|...] ;
C = [case(1, 1), case(1, 2), case(1, 3), case(1, 4), case(1, 5), case(2, 5), case(3, 5), case(3, 6), case(..., ...)|...] ;
C = [case(1, 1), case(1, 2), case(1, 3), case(1, 4), case(1, 5), case(2, 5), case(3, 5), case(3, 6), case(..., ...)|...] ;
false.

?-
```

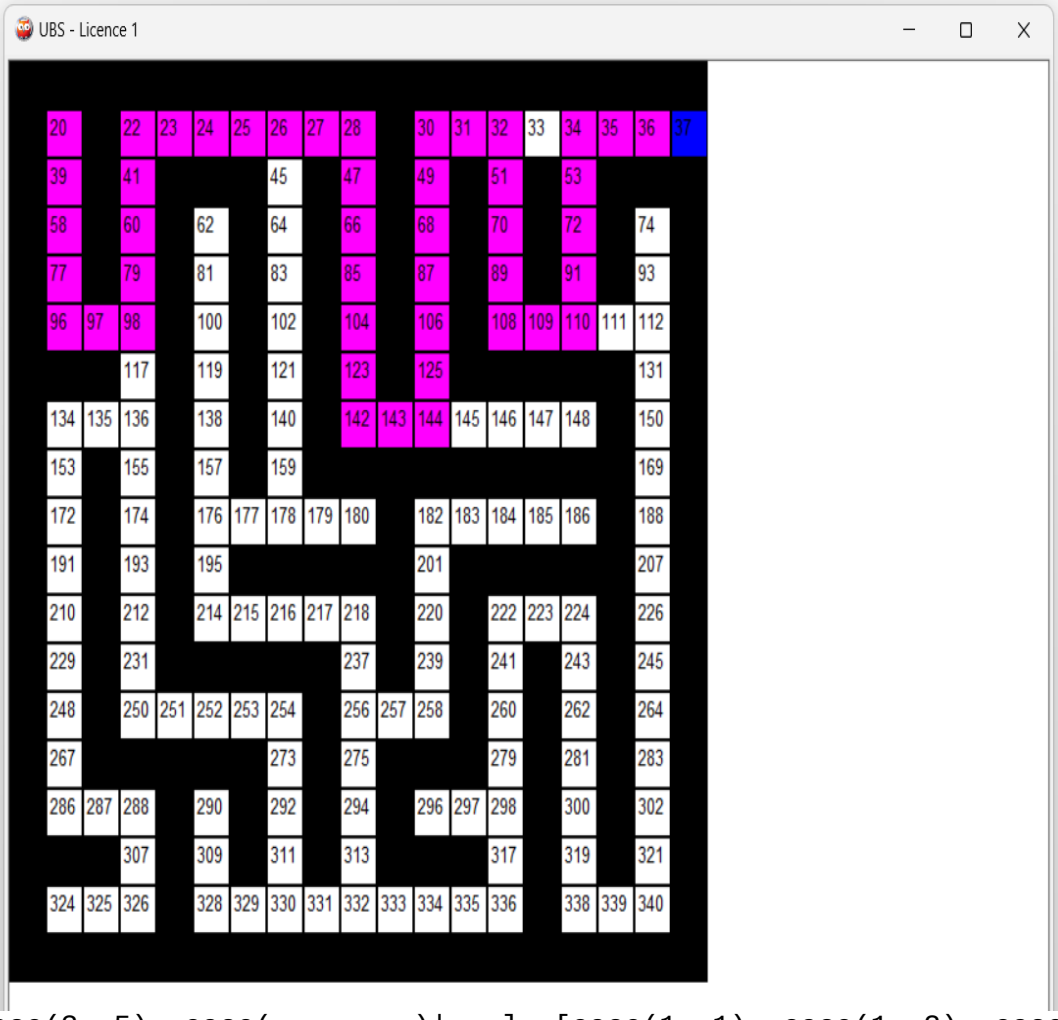
Prédicat eviteMinotaure(X,Y, Chemin)/3

```
eviteMinotaure(X,Y, Chemin) :- avanceDeCollect(1,1,[],Chemin),
    \+ member(case(X,Y),Chemin).
```

Dans la console

```
?- findall(C,eviteMinotaure(13,12,C),L).
L = [[case(1, 1), case(1, 2), case(1, 3), case(1, 4), case(1, 5), case(2,
```

```
?- gr_init,dessine_laby,dessine_laby,avanceDeCollect(1,1,[], Chemin), dessine_chemin(Chemin, magenta), sleep(2),dessine_chemin(Chemin,white),fail.
```




```
5), case(3, 5), case(..., ...)|...], [case(1, 1), case(1, 2), case(1, 3),
case(1, 4), case(1, 5), case(2, 5), case(..., ...)|...], [case(1, 1),
```

```
case(1, 2), case(1, 3), case(1, 4), case(1, 5), case(..., ...)|...],
[case(1, 1), case(1, 2), case(1, 3), case(1, 4), case(..., ...)|...]].

?- findall(C,eviteMinotaure(1,1,C),L).
L = [].

?- findall(C,eviteMinotaure(14,9,C),L).
L = [[case(1, 1), case(1, 2), case(1, 3), case(1, 4), case(1, 5), case(2,
5), case(3, 5), case(..., ...)|...], [case(1, 1), case(1, 2), case(1, 3),
case(1, 4), case(1, 5), case(2, 5), case(..., ...)|...], [case(1, 1),
case(1, 2), case(1, 3), case(1, 4), case(1, 5), case(..., ...)|...],
[case(1, 1), case(1, 2), case(1, 3), case(1, 4), case(..., ...)|...],
[case(1, 1), case(1, 2), case(1, 3), case(..., ...)|...], [case(1, 1),
case(1, 2), case(..., ...)|...], [case(1, 1), case(..., ...)|...],
[case(..., ...)|...]].
```