TD no13	**"Crazy Circus"**

Objectif de la séance :
Résoudre des problèmes d'IA par exploration des solutions à l'aide de Prolog

Crazy circus [1]

Dans le jeu de société Crazy Circus chaque joueur incarne un dompteur s'efforçant de donner les (bons) ordres aux animaux, plus vite que les autres.

Au début du jeu, les trois animaux sont posés au hazard sur les deux podiums (cf. la figure 1(a)). Un tour de jeu consiste à donner des ordres pour obtenir la position d'arrivée indiquée par une carte (cf. la carte présentée par la figure 1(b)). La position d'arrivée des animaux sera la position départ du prochain tour.

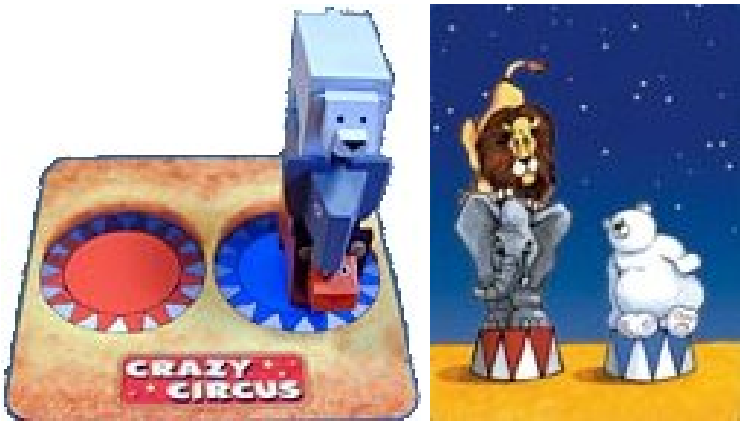
Pour passer de la position de départ à celle d'arrivée, les dompteurs doivent trouver la bonne séquence d'ordres. Il y a cinq ordres possibles : *so*, *ma*, *ni*, *ki* et *lo*. Chaque ordre correspond à un déplacement, comme indiqué dans le tableau 1.

	so : l'animal au sommet du podium rouge échange sa place avec l'animal au sommet du podium bleu.		ma : l'animal en bas du podium rouge va en haut de ce même podium.
	ni : la même chose que MA sur l'autre podium : l'animal en bas du podium bleu va en haut de ce même podium.		ki : l'animal en haut du podium bleu va au sommet du podium rouge.
	lo : la même chose que KI sur l'autre pordium : l'animal en haut du podium rouge va au sommet du podium bleu.		

TABLE 1 – Signification des ordres

Par exemple la séquence d'ordres *ni, ni, ki, ki* permet de résoudre le problème présenté par la figure 1.

1. Règles du jeu issues de http ://crazycircus.free.fr/



(a) Position de départ (b) Position d'arrivée

FIGURE 1 – Le jeu Crazy Circus

Une IA imbattable

L'objectif de cet exercice est de développer des prédicats utilisés par le prédicat *meilleures_solutions/3* tel que `meilleures_solutions(PlateauDepart, PlateauArrive, MeilleuresSolutions)` est vrai lorsque `MeilleuresSolutions` est la liste des suites de longueurs minimales d'ordres pour passer de `PlateauDepart` (décrivant les deux plots rouge et bleu) à `PlateauArrive`.

Par exemple :

```
?- meilleures_solutions(plateau([],[ours, elephant, lion]), plateau([lion,
    elephant], [ours]), Solutions).
Solutions = [[ni, ni, ki, ki], [ki, so, ni, ki], [ki, so, ki, so], [ki, ki, ma, so
    ]].
```

Comme le montre l'exemple ci-dessus, on représente l'état du jeu à l'aide du terme complexe `plateau` d'arité 2 tel que le premier élément est la liste des animaux du plot rouge, et le deuxième la liste des animaux du plot bleu. Pour un plot donné, l'animal se trouvant en tête de liste est l'animal qui est le plus haut sur le plot.

Vous compléterez le fichier `crazyCircus.pl` disponible sur moodle avec les prédicats suivants :

1. Donnez les cinq clauses du prédicat *ordre/1* qui définissent sous forme de faits Prolog les cinq ordres possibles :

   ```
   ?- ordre(Ordre).
   Ordre = so ;
   Ordre = ma ;
   Ordre = ni ;
   Ordre = ki ;
   Ordre = lo.
   ```

2. Donnez les clauses du prédicat *donner_un_ordre/3* tel que `donner_un_ordre(PlateauDepart, PlateauArrive, Ordre)` est vrai lorsque l'ordre `Ordre` permet de passer de `PlateauDepart` à `PlateauArrive` :

   ```
   ?- donner_un_ordre(plateau([],[ours, elephant, lion]), PlateauArrive, ni).
   PlateauArrive = plateau([], [lion, ours, elephant]) ;
   false.
   ?- donner_un_ordre(plateau([],[ours, elephant, lion]), plateau([], [lion,
       ours, elephant]), Ordre).
   Ordre = ni ;
   false.
   ```

3. Donnez les clauses du prédicat récursif *resoudre/4* tel que `resoudre(PlateauDepart, PlateauArrive, Plateaux, Ordres)` est vrai lorsque la liste des `Ordres` permet de passer de `PlateauDepart` à `PlateauArrive` et tel que `Plateaux` représente tous les plateaux produits (cette variable va éviter au Prolog de tourner en rond et, dans le question, elle aura pour valeur la liste contenant le plateau de départ) :

```
?- resoudre(plateau([],[ours, elephant, lion]), plateau([lion, elephant], [
    ours]), [plateau([],[ours, elephant, lion])], ListeDOrdres).
ListeDOrdres = [ni, ni, ki, so, ni, so, ni, so, ki|...] ;
ListeDOrdres = [ni, ni, ki, so, ni, so, ni, so, ki|...] ;
ListeDOrdres = [ni, ni, ki, so, ni, so, ni, so, ki|...]
...
```

4. Donnez les clauses du prédicats *resoudre/3* tel que `resoudre(PlateauDepart, PlateauArrive, Ordres)` est vrai lorsque la liste des `Ordres` permet de passer de `PlateauDepart` à `PlateauArrive` :

```
?- resoudre(plateau([],[ours, elephant, lion]), plateau([lion, elephant], [
    ours]),  ListeDOrdres).
ListeDOrdres = [ni, ni, ki, so, ni, so, ni, so, ki|...] ;
ListeDOrdres = [ni, ni, ki, so, ni, so, ni, so, ki|...] ;
...
```

5. Donnez la clause du prédicat *solutions/3 tel que* `solutions(PlateauDepart, PlateauArrive, Solutions)` est vrai lorsque `Solutions` est la liste de toutes les solutions permettant de passer de `PlateauDepart` à `PlateauArrive` :

```
?- solutions(plateau([],[ours, elephant, lion]), plateau([lion, elephant],
    [ours]),  Solutions).
Solutions = [[ni, ni, ki, so, ni, so, ni, so|...], [ni, ni, ki, so, ni, so,
     ni|...], [ni, ni, ki, so, ni, so|...], [ni, ni, ki, so, ni|...], [ni,
    ni, ki, so|...], [ni, ni, ki|...], [ni, ni|...], [ni|...],
    [...|...]|...].
```

6. Expliquez la déclaration du prédicat *listesDesListes_longueur_listesDeCetteLongueur/3*
7. Expliquez la déclaration du prédicat *meilleurs_solutions/3*