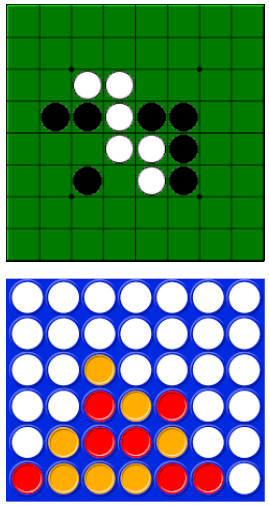
# ALIA 4IF – Septembre 2021

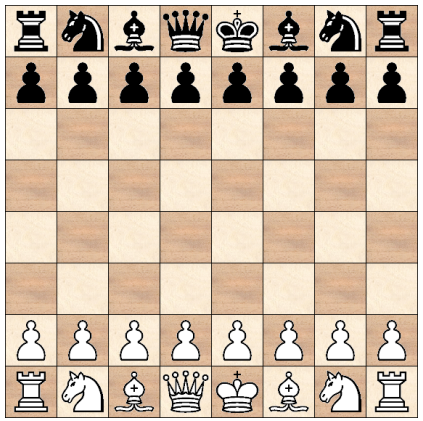
- Objectifs pédagogiques

"Découvrir Prolog : un vecteur de la programmation en logique ; Un outil pour le développement de programmes intelligents"

- Projet : programmation en Prolog d'un jeu à 2 joueurs avec implementations de différentes heuristiques ; Evaluation empirique et rendu sous la forme d'une démonstration par hexanôme

- Quel jeu ?





# Jouons avec un programme simple et idiot ;-)

```
?- init.

New turn for:x
*---------*
???
???
???
*---------*
New turn for:o
*---------*
???
x??
???
*---------*
New turn for:x
*---------*
???
x?o
???
*---------*
New turn for:o
*---------*
```

```
*---------*
x??
x?o
???
*---------*
New turn for:x
*---------*
x?o
x?o
???
*---------*
New turn for:o
*---------*
x?o
xxo
???
*---------*
New turn for:x
*---------*
x?o
xxo
?o?
*---------*
```

```
New turn for:x
*---------*
x?o
xxo
?o?
*---------*
New turn for:o
*---------*
xxo
xxo
?o?
*---------*
Game is Over.
Winner: o
*---------*
xxo
xxo
?oo
*---------*
true.
```

# Tic-Tac-Toe aka Morpion (1)

```
% The game state is represented by a list of 9 elements
% board([_,_,_,_,_,_,_,_,_]) at the beginning
% e.g., board([_,_,'x',_,_,_,_,_,_]) after the first round
% e.g., board([_,_,'x',_,_,_,_,’o’,_]) after the second round
% ... until someone wins or the board is fully instanciated


:- dynamic board/1.


% Test is the game is finished
gameover(Winner) :- board(Board), winner(Board,Winner), !.
gameover('Draw') :- board(Board), isBoardFull(Board).
```

# Tic-Tac-Toe aka Morpion (2)

```
% Test if a Board is a winning configuration for player P.
winner(Board, P) :- Board = [P,Q,R,_,_,_,_,_,_], P==Q, Q==R, nonvar(P).
winner(Board, P) :- Board = [_,_,_,P,Q,R,_,_,_], P==Q, Q==R, nonvar(P).
winner(Board, P) :- Board = [_,_,_,_,_,_,P,Q,R], P==Q, Q==R, nonvar(P).
winner(Board, P) :- Board = [P,_,_,Q,_,_,R,_,_], P==Q, Q==R, nonvar(P).
winner(Board, P) :- Board = [_,P,_,_,Q,_,_,R,_], P==Q, Q==R, nonvar(P).
winner(Board, P) :- Board = [_,_,P,_,_,Q,_,_,R], P==Q, Q==R, nonvar(P).
winner(Board, P) :- Board = [P,_,_,_,Q,_,_,_,R], P==Q, Q==R, nonvar(P).
winner(Board, P) :- Board = [_,_,P,_,Q,_,R,_,_], P==Q, Q==R, nonvar(P).
% Check if all the elements of the List (the board) are instanciated
isBoardFull([]).
isBoardFull([H|T]):- nonvar(H), isBoardFull(T).
```

# Tic-Tac-Toe aka Morpion (3)

% Artificial intelligence: choose in a Board the index to play for Player (_). This "AI" plays randomly and does not care who is playing: it chooses a free position in the Board (an element which is an free variable).

```
ia(Board, Index,_) :-
        repeat, Index is random(9), nth0(Index, Board, Elem), var(Elem), !.
```

```
?- nth0(4,[a,b,c,d,e,f],d).
false.
?- nth0(X,[a,b,c,d,e,f],e).
X = 4 ;
false.
```

```
?- nth0(4,[a,b,c,d,e,f],e).
true.
?- nth0(4,[a,b,c,d,X,f],e).
X = e.
```

Some intelligence can be used ;-)

e.g., Building the tree of the possible moves at a given depth

Min-Max algorithm (extended with the alpha-beta cut mechanism) …

# Tic-Tac-Toe aka Morpion (4)

```
% Game is over, we cut to stop the search, and display the winner.
play(_) :-      gameover(Winner), !, write('Game is Over. Winner: '),
                writeln(Winner), displayBoard.
% The game is not over, we play the next turn
play(Player) :- write('New turn for:'), writeln(Player),
                board(Board),
                displayBoard,
                ia(Board, Move, Player),
                playMove(Board, Move, NewBoard, Player),
                applyIt(Board, NewBoard),
                changePlayer(Player,NextPlayer),
                play(NextPlayer).
```

# Tic-Tac-Toe aka Morpion (5)

```
% Play a Move, the new Board will be the same, but one value will be instanciated with the Move
playMove(Board,Move,NewBoard,Player) :-
        Board=NewBoard, nth0(Move,NewBoard,Player).

% Remove old board - save new on in the knowledge base
applyIt(Board,NewBoard) :-
        retract(board(Board)), assert(board(NewBoard)).

% Predicate to get the next player
changePlayer('x','o').
changePlayer('o','x').
```

# Tic-Tac-Toe aka Morpion (6)

```
% Print the value of the board at index N (?, x or o)
printVal(N) :- board(B), nth0(N,B,Val), var(Val), write('?'), !.
printVal(N) :- board(B), nth0(N,B,Val), write(Val).

% Display the board
displayBoard:-  writeln('*----------*'),
                printVal(0), printVal(1), printVal(2), writeln(''),
                printVal(3), printVal(4), printVal(5), writeln(''),
                printVal(6), printVal(7), printVal(8), writeln(''),
                writeln('*----------*').
% Start the game!
init :- length(Board,9), assert(board(Board)), play('x').
```