**GUIÓN DE PRÁCTICAS**

## BÚSQUEDA ENTRE ADVERSARIOS: MINIMAX

El código facilitado corresponde a una implementación (aún incompleta) de una versión del 3 en Raya, el juego TicTacToe, donde el objetivo es conseguir 3 fichas iguales en la misma fila, columna o diagonal.
A diferencia del 3 en Raya, las fichas se van colocando en el tablero, pero no se desplazan dentro de él, por lo que el número total de fichas es 9.
El juego finaliza cuando un jugador ha conseguido 3 fichas iguales en las posiciones correctas o cuando el tablero se completa sin que ningún jugador haya ganado (empate).

Dispones de 3 archivos con código fuente: tictactoeAlum.py con la formalización del problema, minimaxAlum.py con las funciones necesarias para implementar las estrategias de búsqueda para juegos de 2 adversarios de dos adversarios y main.py

1. Abre el archivo main.py y familiarízate con el código que aparece para controlar el bucle principal del juego. Intenta ejecutarlo.
2. Analice las funciones de minimaxAlum.py
3. Completa la formalización del juego del TicTacToe. En concreto, implementa las funciones de aplicaJugada, esValida, terminal y utilidad.
4. Pruebe a ejecutar ahora main.py. ¿Funciona tal y como esperaba?
5. Completa la implementación de la estrategia minimax para que todo funcione correctamente y se pueda jugar al TicTacToe con el Agente Inteligente. Para ello, use el pseudocódigo que se le ofrece a continuación.

```
tNodo: función minimax(E Nodo: nodo)
var
  entero: jugador = 1
  Jugada: mejorJugada = jugadas[0]
inicio
  max <- -10000
  para cada jugada en jugadas hacer
    si esValida(nodo, jugada) entonces
      intento <- aplicaJugada(nodo, jugada, jugador)
      max_actual <- valorMin(intento)
      si max_actual > max entonces
        max <- max_actual
        mejorJugada <- jugada
  nodo=aplicaJugada(nodo, mejorJugada, jugador)
  devolver nodo
```

**GUIÓN DE PRÁCTICAS**

```
entero: función valorMin(E Nodo: nodo)
var
   entero: valor_min = ∞, jugador=-1
inicio
   si terminal(nodo) entonces
      valor_min <- utilidad(nodo)
   si_no
      valor_min <- ∞
      para cada jugada en jugadas hacer
         si esValido(nodo, jugada) entonces
            valor_min <- min(valor_min,
                        valorMax(aplicaJugada(nodo, jugada, jugador)));
   devuelve valor_min
```

```
entero: función valorMax(E Nodo: nodo)
var
   entero: valor_max = -∞, jugador=1
inicio
   si terminal(nodo) entonces
      valor_max <- utilidad(nodo)
   si_no
      valor_max <- -∞
      para cada jugada en jugadas hacer
         si esValida(nodo, jugada) entonces
            valor_max <- max(valor_max,
                        valorMin(aplicaJugada(nodo, jugada, jugador)));
   devuelve valor_max
```