In the Name of God

Sharif University of Technology - Department of Computer Engineering

Artificial Intelligence - Mr. Samiei

Spring 2023

Practical Assignment(Adversarial Search)

Deadline: Esfand 19th - 23:59
Cheating is Strongly Prohibited

Please run all the cells.

Personal Info

Student number = 99105678 Name = Kiarash Last_Name = Kianian

Libraries

In [1]:
```
! pip install pygame
! pip install numpy
```

```
Requirement already satisfied: pygame in c:\programdata\anaconda3\lib\site-pa
ckages (2.2.0)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-pac
kages (1.20.1)
kkkkk
```

Adversarial Search (50 Points)

Author: Erfan Sadraiye
Please run all the cells.

In this problem you must implement a minimax agent to play a simple game know as Tic-tac-toe.

XO is a paper-and-pencil game for two players who take turns marking the spaces in a three-by-three grid with X or O. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row is the winner.

The game has already been implemented for you. You should implement two things:
1 - Implement the two minimax player logic in *Player.py* . For the first player you must create the minimax tree until you reach a terminal node(a state which a player has won or there is a draw) or a cutoff state. Then you must use an evaluation function to score the leafs and find the next best move. For the second player you must do all the above however when you are playing as max there is a probability that you will choose a random move instead of the best move. 2 - Implement the evaluation function in *Board.py*. Since we can't create the full game tree we need some sort of scoring function to differentiate between states. For example, we can count the pieces that are neighbours. better evaluation function will lead to a better minimax player.

We will use the commented function in main.py to grade your code. You can also test your code by playing with it or using a random player.

You can play the game with a random player by running the next cell. Upon running it a GUI of the game will open. You can also change the players' type in this section.

In [2]:
```
from Game import XO
from Player import HumanPlayer, RandomPlayer, MiniMaxPlayer, MiniMaxProbPlayer

# player1 = RandomPlayer(1)
# player2 = HumanPlayer(2)
# game = XO(player1, player2)
# game.play_game_with_gui()
```

```
pygame 2.2.0 (SDL 2.0.22, Python 3.8.8)
Hello from the pygame community. https://www.pygame.org/contribute.html (https://www.pygame.org/contribute.html)
```

Complete the play function of MiniMaxPlayer and MiniMaxProbPlayer in *player.py* and score_position function in *Board.py*, minimax and minimax_prob functions in *alpha_beta.py* then run the following cells to test your implementation.

For earning full mark, MiniMaxPlayer shouldn't lose against RandomPlayer and MiniMaxProbPlayer, and you should win at least 45 game of these 50 game.

In [33]:
```
from Game import XO
from Player import HumanPlayer, RandomPlayer, MiniMaxPlayer, MiniMaxProbPlayer
def get_game_result(player1, player2, num_game):
    win, lose, draw = 0, 0, 0
    for i in range(num_game):
        game = XO(player1, player2)
        result = game.play_game_without_gui()
        if result == 1:
            win += 1
        elif result == 2:
            lose += 1
        else:
            draw += 1
    return win, lose, draw


def mark():
    player1 = MiniMaxPlayer(1, depth=4)
    player2 = RandomPlayer(2)
    player3 = MiniMaxProbPlayer(2, depth=3, prob_stochastic=0.8)
    win1, lose1, draw1 = get_game_result(player1, player2, 40)
    win2, lose2, draw2 = get_game_result(player1, player3, 10)
    print(f'minimax player vs random player win={win1}, lose={lose1}, draw={dr
    print(f'minimax player vs minimax prob player win={win2}, lose={lose2}, dr
    if win1 + win2 >= 45 and lose1 + lose2 == 0:
        print('you will earn the full mark from random player vs minimax')


mark()
```

```
player1 goes first.
PLAYER 1 WINS!
[[2. 2. 1.]
 [0. 1. 0.]
 [1. 0. 0.]]
player1 is O. player2 is X.
player1 goes first.
PLAYER 1 WINS!
[[2. 2. 1.]
 [0. 1. 0.]
 [1. 0. 0.]]
player1 is O. player2 is X.
player1 goes first.
PLAYER 1 WINS!
[[2. 2. 1.]
 [0. 1. 0.]
 [1. 0. 0.]]
minimax player vs random player win=39, lose=0, draw=1
minimax player vs minimax prob player win=10, lose=0, draw=0
you will earn the full mark from random player vs minimax
```