DS2020 - Artificial Intelligence
Lab 2

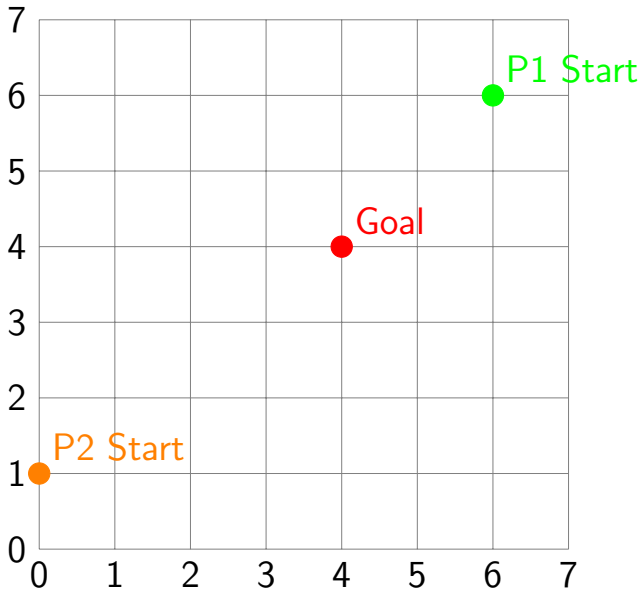
Due on 13/03/2025 11.59pm

Instructions:

- You are expected to follow the honor code of the course while doing this homework.
- **This lab should be completed individually**.
- You will be provided with 3 files **player.py**, **playgame.py** and **adversary.py**. **Do not** make any modifications to **playgame.py** and **adversary.py**
- Rename the **player.py file** using your roll number (e.g., `12345678.py`) before submitting it to Moodle.

Adversarial Yantra Hunt

This assignment is a variation of the Yantra Hunt problem from Lab 1 where we are introducing an adversary. The game has 2 players, Player 1 and Player 2, both of whom compete to reach the goal state first. Your objective is to write the code for Player 1, so that it reaches the goal state before Player 2(the Adversary).
An example grid setup is illustrated below:



Game Setup(with respect to the sample grid)

- The game is played on a 7x7 grid.
- Player 1 and Player 2 start from predefined positions.
- The goal position is (4,4).
- The game alternates turns between Player 1 and Player 2, with Player 1 starting the game.
- Player 2 will follow either a **novice** or **amateur** strategy while moving.
- Allowed Moves: 1 step North (N), South (S), East (E), or West (W) per move.
- **The game ends when a player reaches the goal or a draw condition occurs.**

Constraint During a player's turn, whatever move they make, the opponent is forced to mimic it. The reverse is also true when it is the opponent's turn. Hence, the objective of your move is to maximize your(Player 1) chances at winning while minimizing the opponent's(Player 2's) chances. For example:-

- **Initial Positions:** Player 1 is at (6,6), Player 2 is at (1,0) and Goal is at (4,4).
- **Player 1's Turn:** Player 1 moves 1 step west. Player 2 will also be moved 1 step west. However, since it is not a valid move, Player 2 stays where it is.
- **Player 2's Turn:** Player 2 moves 1 step east. Player 2 will also be moved 1 step east.

Assignment Setup and Implementations:

The accompanying zip file includes the lab starter code. The `playgame.py` file contains the code for grid setup with the goal state and the initial positions of both players *(you can use it to test your code)*, while the `adversary.py` file contains the code for Player 2 for both Novice and Amateur scenarios. **You can run playgame.py file to test your output**

You are required to implement the following functions only in the `player.py` file. Do not make any other modifications.

- **is_valid(self, pos)**: Checks if the position is within the grid boundaries.

- **move_player(self, player, direction)**: Moves the specified player if the move is valid.
- **utility(self, pos)**: Computes a utility score based on the player's distance to the goal vs. the opponent's distance to the goal.
- **best_player_move(self, pos)**: Determines the best move for Player 1 as per the current strategy for Player 1(obtained from self.player_strat).
- **random_move(self, pos)**: Chooses a random **valid** move from your immediate neighbor cell(i.e. move one step N/S/E/W). . You can use this function to familiarize yourself with how the game will played between the players.
- **minimax_vanilla(self, pos, depth, max_turn)**: Determines the best move using vanilla Minimax algorithm (without alpha-beta pruning).
- **minimax(self, pos, depth, alpha, beta, max_turn**: Implements the minimax algorithm with alpha-beta pruning to determine the best move.

Evaluation: Your code will be evaluated with additional positions for the players/Goal state, and different grid sizes. Full marks will be awarded if your code correctly predicts the game outcome **(Player1 Wins/ Player2 Wins/ Draw)** along with the path taken by Player 1.