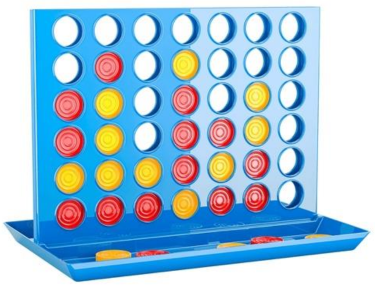
3rd Assignment : Implementing Connect Four Using Adversarial Search



Objective

In this assignment, you will implement the classic game ***Connect Four*** using adversarial search techniques, specifically the Minimax algorithm with **Alpha-Beta pruning**.

-The goal is to create an AI opponent that can play optimally against a human player .

Game Description: Connect Four

Connect Four is a two-player board game played on a 6-row by 7-column vertical grid. Players take turns dropping discs from the top into one of the seven columns. The disc falls straight down, occupying the lowest available space in the column. The first player to form a horizontal, vertical, or diagonal line of **four** of their own discs wins.

What to Implement

- Your task is to **build the game from scratch** using your programming language of choice.

Technical Requirements

You will implement the following components:

1. Game Board Representation

- Use a **2D array (6x7)** to represent the board.
- Initialize the board as empty (**None** or **0** for empty, **1** for Player 1, **2** for Player 2).

2. Game Mechanics

- **Valid Moves:** Ensure a disc is placed in a valid column (not full).
- **Winning Condition:** Check after each move if a player has connected four discs.
- **Draw Condition:** Check if the board is full without a winner.

3. Adversarial Search (AI Opponent)

- Implement the **Minimax algorithm** with **Alpha-Beta pruning** to make optimal moves.

- **Define a heuristic utility function** to score non-terminal game states **(NO similar Heuristic function are allowed between students)**
- Set a **depth limit** (e.g., depth=5) to ensure the AI runs efficiently.

  You can use IDDFS if you will set a timer for players (extra grades)

4. User Interface

- A simple **text-based UI** (printing the board in the **console**).
- (Extra grade) A **graphical UI**

Assignment Structure (suggested for Python)

connect_four/

```
game.py         # Core game logic (board state, moves, win check)
player.py       # Base player classes (Human, AI)
minimax.py        # Minimax + Alpha-Beta pruning implementation
heuristic.py     # heuristic utility evaluation  functions
main.py         # Game loop and user interface (CLI or GUI)
```