1. **You are free to choose your own static evaluation function. Justify your choice of static evaluation value design and explain with a sample game state. Do not use any machine learning model for the evaluation function.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

EVALUATION FUNCTION

A static evaluation function is required for non-terminal states (since Minimax will expand possible moves up to some depth).

The key idea

* Reward AI (T) for creating sequences of 2, 3, or 4 in a row / column / diagonal
* Penalize Human (X) for creating similar sequences.
* The closer a player is to 4 in a row, the higher the score.

Design

* **Winning State:**  
  If the AI wins → +∞ (large positive, e.g., +1000)  
  If the human wins → -∞ (large negative, e.g., -1000)
* **Intermediate State:**  
  Assign scores based on sequences of consecutive marks:
  + 3 in a row (open on both ends): +100 if AI, -100 if human
  + 2 in a row (open): +10 if AI, -10 if human

This ensures:

* AI prioritizes winning.
* AI blocks human’s winning threats.
* AI strategically builds up to 4-in-a-row.

Sample Game State

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **X** |  | **T** | **T** |  |
| **X** | **T** | **T** |  |  |
|  | **X** |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Step 1 : Check sequences

1. Row 0 -> TT -> AI gets +10
2. Row 1 -> TT -> AI gets +10
3. Column 0 -> XX -> Human gets -10
4. Diagonal (1,0 -> 2,1) -> XX -> Human gets -10

Diagonal (1,1 -> 0,2) -> TT -> AI gets +10

Diagonal (1,2 -> 0,3) -> TT -> AI gets +10

Step 2 : Total Evaluation

E = +10 + (+10) + (-10) + (-10) + (+10) + (+10)

= +20

So this state slightly favors the AI

**B. Similar to the virtual lab example, one of the players must be a human ie., it must get dynamic inputs from us. The other player must be simulated using the program.**

1. Human to provides input (row, col for their move).

2. AI uses Minimax (with alpha-beta pruning for efficiency).

3. The program alternates turns until a win/draw.