This is a league based challenge.

For this challenge, multiple leagues for the same game are available. Once you have proven yourself against the first Boss, you will access a higher league and harder opponents will be available.

NEW: In wood leagues, your submission will only fight the boss in the arena. Win a best-of-five to advance.

Goal

End the game with a higher **score** than your opponent.
Three players are pitted against one another in the **arcade olympics**.

Each player controls a character in **four** mini-games **simultaneously**. Earn a maximum of **medals** in all four games to acquire the highest **score**.

Rules

Each player is hooked up to **four** different arcade machines, and each of these machines is running the Hurdle Race **mini-game**. Your code can read the 8 **registers** used internally by the machines: `GPU`, containing a string and `reg0` to `reg6` containing integers. What those values represent is different for each game.

The game is played in turns. On each turn, all three players perform one of four possible actions: `UP`, `DOWN`, `LEFT`, or `RIGHT`.

When an action is performed by a player, their agent in **each** mini-game performs that same action, because the controls have been wired to all 4 machines at once.

Earning medals

The four mini-games play on loop throughout the game. In each run of a mini-game you may acquire a gold, silver or bronze **medal**. In between runs is a **reset** turn where the mini-game is inactive.

At the end of the game, each player's score for each mini-game is calculated based on the number of medals earned in total, with this formula:

mini_game_score = nb_silver_medals + nb_gold_medals * 3

The scores for all **four** mini-games are **multiplied together** to determine the **final score**.

During a reset turn, the `GPU` register will show `"GAME_OVER"`.

If there are ties in a mini-game, tied players will win the same highest medal. For instance, if two players tie for first place, they will both win gold and the third player will receive **bronze**.

Mini-game 1: Hurdle Race

This mini-game is a race between the three agents. Each agent is on the same randomly generated race track. The racetrack is composed of **30 spaces**, agents start on the first space, and the last space is the finish line. A space may contain a **hurdle** which the agents must **jump** over or else they will **collide** with it and be **stunned** for the next `3` turns. A stunned agent will not move regardless of the action performed.

On each turn, the agents can perform one of the following actions:

- `UP`: jump over one space, ignoring any hurdle on the next space and moving by `2` spaces total.
- `LEFT`: move forward by `1` space.
- `DOWN`: move forward by `2` spaces.
- `RIGHT`: move forward by `3` spaces.

Moving into a hurdle will interrupt the agent's movement, stopping on the same space as the hurdle.

When either agent reaches the **finish**, the run ends. The players are awarded a medal based on their positions in the race, and the next run begins after a **reset** turn.

Register	Description	Example
GPU	ASCII representation of the racetrack. `.` for empty space. `#` for hurdle.	`.....#...#...#................`
reg0	position of player 1	0
reg1	position of player 2	6
reg2	position of player 3	12
reg3	stun timer for player 1	1
reg4	stun timer for player 2	0
reg5	stun timer for player 3	2
reg6	*unused*	

The **stun timer** is the number of turns remaining of being stunned (`3`, then `2`, then `1`). `0` means the agent is not stunned.

Victory Condition

You have a higher **final score** after `100` turns.

Defeat Condition

Your program does not provide a command in the allotted time or it provides an unrecognized command.

Debugging tips

- Press the gear icon on the viewer to access extra display options.
- Use the keyboard to control the action: space to play/pause, arrows to step 1 frame at a time.

Game Protocol

Initialization Input

First line: `playerIdx` an integer to indicate which agent you control in the mini-games.
Next line: the number of simultaneously running mini-games. For this league it's `4`.

Input for One Game Turn

Next `3` lines: one line per player, ordered by `playerIdx`. A string `scoreInfo` containing a breakdown of each player's final score. It contains `13` integers. The first integer representing the player's current **final score points** followed by three integers per mini-game: `nb_gold_medals`, `nb_silver_medals`, `nb_bronze_medals`.

Next `nbGames` lines: one line for each mini-game, containing the eight space-separated registers:

- gpu a string
- reg0 an integer
- reg1 an integer
- reg2 an integer
- reg3 an integer
- reg4 an integer
- reg5 an integer
- reg6 an integer

Unused registers will always be -1 .

Output

One of the following strings:

- `UP`
- `RIGHT`
- `DOWN`
- `LEFT`

Constraints

0 ≤ playerIdx ≤ 2
1 ≤ nbGames ≤ 4 *(across all leagues)*

Response time per turn ≤ 50 ms
Response time for the first turn ≤ 1000 ms

What is in store for me in the higher leagues?

- 4 entirely different mini-games will be played simultaneously!