This is a league based challenge.

For this challenge, multiple leagues for the same game are available. Once you have proven your skills against the first Boss, you will access a higher league and extra rules will be available.

NEW: In wooden leagues, your submission will only fight the boss in the arena. Complete the objective specified in each league at least 3 times out 5 to advance to the next league.

Goal

Win the water fight by controlling the most territory, or out-soak your opponent!

Rules

The game is played on a **grid**.

Each player controls a team of **agents**.

Each turn, every agent can perform one **move action** and/or one **combat action**.

Agents

Agents are the player-controlled units on the field. They possess **attributes** and actions.

Each agent has a `wetness` meter, which goes up when getting attacked by enemy agents. Once an agent's `wetness` reaches 100, they are eliminated and removed from play.

Agents also have a set `soaking_power` and `optimal_range`. The power indicates how much wetness they normally deal, while the range is used to apply a penalty if the target is too far.

- Up to the `optimal_range`, **shooting** deals 100% of their `soaking_power`.
- Beyond that, and up to twice the `optimal_range`, **shooting** only deals 50% of their `soaking_power`.
- Beyond that, the shot fails.

Note: All distances are calculated with the Manhattan formula.

Each agent also has a `shoot_cooldown`, which is the amount of turns they must wait after **shooting** to be able to `SHOOT` again. They can still use other actions in the meantime.

In addition to shooting, each agent has a set amount of splash `bombs` that they can throw. The amount is determined at the start of the game and can differ between agents.

Actions

On each turn, you must output one command for each agent that you control. Each command can contain several actions; at most one **move action** and one **combat action**.

You can instruct the actions in any order you want, but the execution order will depend on each action's priority; see the **Action order per turn** section for more details.

Moving is done with the `MOVE x y` command. With it, the agent will attempt to move to the location x, y. If the target location is not orthogonally adjacent to the agent, then they will attempt to move towards it using the shortest valid path possible. If the action results in a movement on a tile with a cover or another agent on it, the movement will be cancelled. If agents collide while attempting to `MOVE`, their movement will be cancelled.

There are several combat actions available:

- `SHOOT id`: Attempt to shoot agent `agentId`. This will deal `wetness` according to the `optimalRange` and `soakingPower` of the agent, and is reduced by any damage reduction gained by the target agent (see the `HUNKER_DOWN` action and the **Cover** section).
- `THROW x y`: Attempt to throw a **splash bomb** at the location `x`, `y`. **Splash bombs** can only be thrown at a maximum distance of **4** tiles away from the agent. They deal **30** `wetness` to agents on the tile it lands on and to all adjacent tiles (orthogonally and diagonally). This action **ignores** damage reduction from covers & hunkering.
- `HUNKER_DOWN`: Hunker down to gain 25% damage reduction against enemy shots this turn. This can be stacked with cover bonuses (see the **Cover** section below).

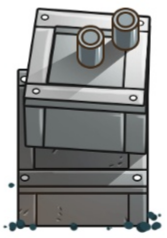
See the Game Protocol section for more information on sending commands to your agents.

Cover

Each tile of the **grid** is given to your program through the standard input. For each column of each row you are given a `tileType`. It can now have one of **three** possible values:

- **0** an empty tile.
- **1** a tile containing **low cover.**
- **2** a tile containing **high cover.**

Tiles with **cover** are impassable, and agents will automatically path around them when perform a `MOVE` action.



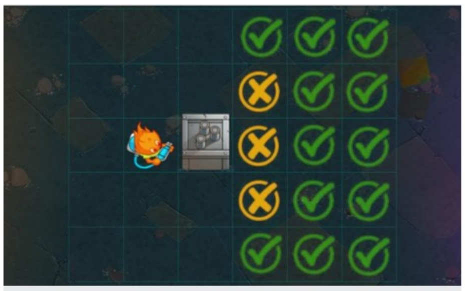
An agent that benefits from a cover will gain damage reduction against enemy shots. Low Covers provide **50%** protection, and High Covers provide **75%** protection.

For instance, an agent within optimal range and a soaking power of 24 will only deal 6 wetness to an enemy behind High Cover.

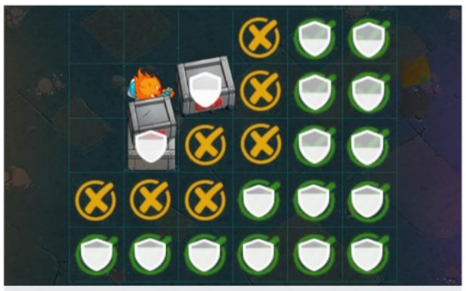
To benefit from a cover, the agent must be orthogonally adjacent to it, and the enemy shot must come from the opposite side of the cover tile. The cover is ignored if both agents are adjacent to the cover.

In the case where multiple covers can be considered, only the **highest** cover will count.

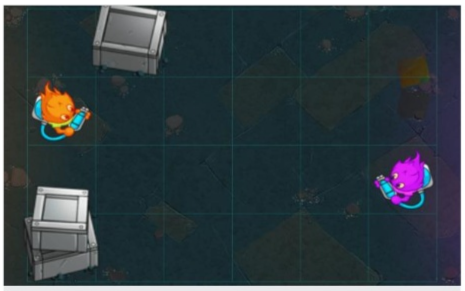
Examples:



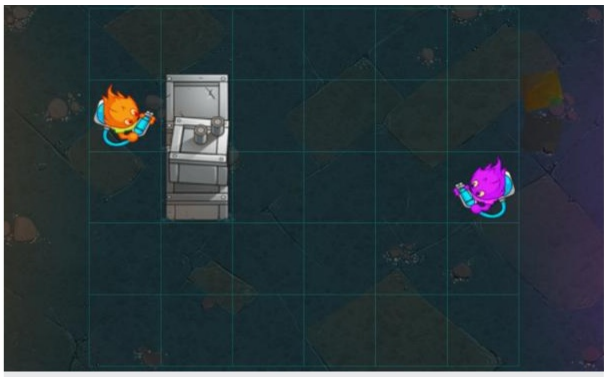
An agent orthogonally adjacent to the left side of a low cover. Shooting this agent from a green tile will result in damage reduction.

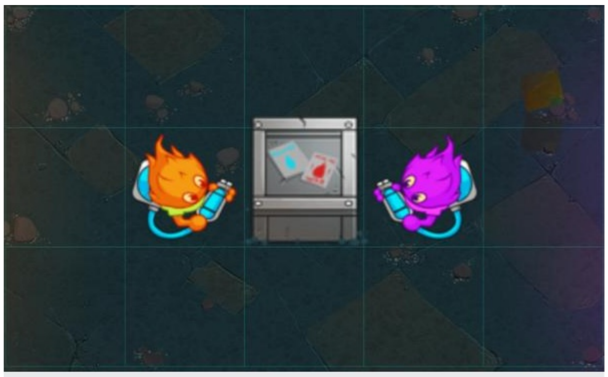


In the case where a shot can be affected by two covers, only the highest one counts.



An agent that is not orthogonally adjacent to any cover, thus not benefitting from their damage reduction.


The orange agent benefits from a low cover while the purple agent does not benefit from any cover.


Neither of these agents benefit from the cover from each other since they are both adjacent to it.

Note: Hunkering down stacks with cover, which means the total protection gained from both taking cover and hunkering down is 75% or 100%, for low and high cover respectively.

📈 Points

You gain points by taking control of tiles when you control a larger area than your opponent.

Any tile that is closer to one of your agents than to an enemy agent is considered to be under your control. However, if an agent has `wetness` greater or equal to **50**, the distance to that agent will be **doubled** during this comparison.

Each turn, if you control **more** tiles than your opponent, you score as many points as extra tiles that you control compared to your opponent.

🎬 Action order for one turn

Game turns are synchronous, for both players and agents (meaning all agents perform their actions at the same time). However, some actions have priority over others:

- `MOVE` actions go first,
- Then `HUNKER_DOWN` actions,
- Then `SHOOT` and `THROW`,
- And finally, the removal of any soaked agent.

Victory Conditions

The winner is the player who fulfills one of the following conditions:

- Reach **600** **more** points than their opponent
- Eliminate all opposing agents
- Have the most points after **100** turns

Defeat Conditions

Your program does not provide a command in the alloted time or one of the commands is invalid.

🐞 Debugging tips

- Hover over the grid to see extra information on the tile under your mouse.
- Assign the special `MESSAGE text` action to an agent and that text will appear above your agent.
- 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.

Click to expand

Game Protocol

Initialization Input

First line: one integer myId, for your player identification.
Second line: one integer agentDataCount for the number of agents on the grid.

Next agentDataCount lines: The following **6** inputs for each agent:

- agentId: unique id of this agent
- player: id of the player owning this agent
- shootCooldown: min number or turns between two shots for this agent
- optimalRange: the optimal shooting range of this agent
- soakingPower: the maximum wetness damage output of this agent
- splashBombs: the starting amount of splash bombs available to this agent

Next line: two integers width and height for the size of the grid.

The next width * height lines: The following 3 inputs for each tile on the grid:

- x: X coordinate (0 is leftmost)
- y: Y coordinate (0 is uppermost)
- tile_type:
  - **0** for an empty tile
  - **1** for a low cover
  - **2** for a high cover

Input for one game turn

First line: one integer agentCount for the number of remaining agents on the grid.

Next agentCount lines: The following **6** inputs for each agent:

- agentId: unique id of this agent
- x: X coordinate (0 is leftmost)
- y: Y coordinate (0 is uppermost)
- cooldown: number of turns left until this agent can shoot again
- splashBombs: current amount of splash bombs available to this agent
- wetness: current wetness of the agent

Next line: one integer myAgentCount for the number of agents controlled by the player.

Output

A single line per agent, preceded by its agentId and followed by its action(s):

Up to one move action:

- `MOVE x y`: Attempt to move towards the location x, y.

Up to one combat action:

- `SHOOT id`: Attempt to shoot agent `agentId`.
- `THROW`: Attempt to throw a splash bomb at the location `x`, `y`.
- `HUNKER_DOWN`: Hunker down to gain 25% damage reduction against enemy attacks this turn.

Up to one message action:

- `MESSAGE text`: Display `text` in the viewer. Useful for debugging.

Instructions are separated by semicolons. For example, consider the following line:

`3;MOVE 12 3;SHOOT 5`

This instructs **agent 3** to **move towards the coordinates (12, 3)** and to **shoot agent 5**.

Note: The `agentId` *at the start can be omitted. In that case, the actions are assigned to the agents in ascending order of* `agentId`.

Constraints

Response time per turn ≤ **50** ms
Response time for the first turn ≤ **1000** ms
12 ≤ `width` ≤ **20**
6 ≤ `height` ≤ **10**
6 ≤ `agentDataCount` ≤ **10**