League of Rockets

An atomic bomb destroyed the Universe. Now it is just a bunch of asteroids floating in space. Your job is to clean them up.

Overview

The game takes place on a plane (with standard Cartesian coordinates), inside of a circle. Two players compete against each other. Each player can control three spaceships. A spaceship can fly and launch rockets. There are also asteroids, which are completely stationary, and your spaceships should avoid them (colliding with an asteroid destroys the spaceship). The goal of the game is to get as many points as possible: you get points by damaging the asteroids and destroying the opposing spaceships.

Spaceships and asteroids are represented by circles. Rockets are represented by points. Destroying a rocket causes an explosion, which is also represented by a circle. Explosions destroy spaceships and deal damage to asteroids. Rocket is destroyed when it comes in contact with an asteroid, an enemy spaceship or an explosion, and also if it flies outside the playfield. There is also a way for a rocket to blow itself up (more on that later). Rockets do not collide with each other.

At any given moment, a spaceship has some acceleration vector, which is applied to its current velocity, which is then used to move the spaceship. At the beginning, a spaceship is stationary and has zero acceleration. Players can apply a different acceleration vector at any time. Additionally, they can stop a spaceship immediately.

Players can tell a spaceship to launch a rocket, but they can't control them directly. In order to launch a rocket, it needs to be programmed. A program is just a sequence of commands for the rocket to execute. There are commands to control the engine power and angular acceleration, you can also blow up the rocket. You can time the execution of commands by issuing delays.

If an explosion produced by one of your rockets comes in contact with an asteroid, you get points. It also decreases the number of hit points of the asteroid - when it reaches zero, the asteroid is destroyed.

Please note that rockets do not explode on contact with allied spaceships, but the explosion itself can destroy any spaceship, enemy or not. Also, only rockets can cause explosions - asteroids and spaceships just disappear.

Detailed description

Gameplay is divided into turns, each turn takes approximately $\frac{1}{FPS}$ seconds (FPS is a server-dependent parameter). Each turn consists of two phases - accepting commands from the players and applying them. Any changes made by the commands will be visible at the start of the next turn.

A lot of matches are taking place at the same time. Each match takes a fixed number of turns (there is no way for it to end early). A player fights with a number of opponents simultaneously. At some point, every player played with every other player exactly once - this is when tournaments ends, and points for all matches are awarded. For every match in a tournament, the arrangement of asteroids and spaceships is the same. Furthermore, it is always centrally symmetric through $(0, 0)$.

Movement

At the end of any turn, each spaceship has some acceleration vector **a** (set by the player), velocity vector **v**, and position **p** (which is a point on a plane). Then new values (visible at the start of the next turn) are calculated as follows:

$$\mathbf{v'} = \mathbf{v} + \mathbf{a}$$

$$\mathbf{p'} = \mathbf{p} + \mathbf{v'}$$

If player did not set any acceleration vector, previous value of the acceleration vector is used. Additionally, a player can choose to stop the spaceship immediately. In that case, velocity is set to 0.

Similarly, each rocket has an accelaration vector **a**, velocity vector **v** and position **p**, and it moves in the same way, but rocket commands do not set the acceleration vector directly. They can set its length to any

number between zero and some maximum length, but its orientation is determined by angle (direction) of the rocket **d** (which is a number in radians, so zero means a direction of the X axis, π is in the opposite direction, and the values increase counter-clockwise). Each rocket also has angular velocity $\mathbf{v}_d$ and angular acceleration $\mathbf{a}_d$, and commands only influence this last value. Other values are then calculated as follows:

$$\mathbf{v'}_d = \mathbf{v}_d + \mathbf{a}_d$$

$$\mathbf{d'} = \mathbf{d} + \mathbf{v'}_d$$

Launching rockets

Each spaceship can launch rockets. In order to do that, you have to provide it with a sequence of commands. Possible commands are as follows:

- `SET_ENGINE` – set the length of the acceleration vector of the rocket,
- `SET_THRUSTERS` – set the angular acceleration of the rocket,
- `EXPLODE` – blow up the rocket,
- `DELAY` – delay the execution of commands further in sequence.

Other than that, you have to provide starting velocity $\mathbf{v}$ and angular velocity $\mathbf{v}_d$. Rocket is spawned at the center of the spaceship. Every command apart from `DELAY` takes one turn to execute - in the next turn, next command from the sequence is executed. You specify how many turns `DELAY` takes to execute (other than that it does nothing). Once the last command is reached, no more commands are executed.

Once the rocket explodes (either by colliding with an object, by executing a command or by flying outside the playfield), an explosion will be present on the next turn. An explosion is just a circle, and its center will be at the position of the exploded rocket. Explosion lasts only for one turn.

There is a limit for a number of commands in the sequence. You can't launch rockets too often - after launching one, you have to wait for some number of turns before you launch another one with the same spaceship.

Asteroids

Asteroids are stationary circles. Every asteroid has some radius, number of hit points and number of points you get for hitting it with an explosion (these values can be different for different asteroids). Asteroids can overlap.

When an explosion overlaps with an asteroid, its hit points decreases by one, and player that caused the explosion gets points. If at the end of the turn an asteroid has zero or less hit points, it is destroyed and removed from the map.

Spaceships

Each spaceship is a circle, and every spaceship has the same radius. If at any point a spaceship comes in contact with an asteroid, an explosion, other spaceship or it will fly outside the big circle, it will be instantly destroyed. If this was caused by an explosion cause by enemy rocket, the enemy gets points. Note that spaceship is not destroyed by colliding with an enemy rocket, but when it happens it triggers an explosion on the next turn, which can destroy the spaceship.

Order of events

After all commands from both players are gathered, events are happening in the following order:

- Acceleration vectors of the spaceships provided by players are set. Rockets launched by the players are spawned.
- Rocket commands are executed (angular accelerations are set, lengths of acceleration vectors are modified, rockets are destroyed - an explosion will appear on the next turn).
- New velocities, positions, angular velocities and angles are calculated and set.
- All collisions are resolved at once:

  - If any spaceship overlaps with any explosion, spaceship is set to be destroyed. If the explosion belonged to an enemy, he gets points.
  - If any spaceship overlaps with an asteroid, any other spaceship or is not completely inside the playfield, it is set to be destroyed.
  - If any asteroid overlaps with any explosion, it loses one hit point for each explosion. Owner of the explosion gets points.
  - If any rocket overlaps with any explosion, asteroid, enemy spaceship, or flies out of the playfield, it is destroyed (an explosion will be appear on the next turn).
- Asteroids with zero or less hitpoints disappear.
- Spaceship set to be destroyed disappear.
- Existing explosions disappear. Explosions from destroyed rockets get created.

Commands

All coordinates, lengths, accelerations and velocities are real (floating-point) numbers. Every such number is printed with 5 decimal digits. Other numbers are integers.

WAIT

Immediately responds with `OK` (like every command), then waits until the start of the next turn, and returns `OK` again.

```
> WAIT
< OK
(after some time...)
< OK
```

GET_ARENA

Returns the number of simultaneous matches you're currently playing, and your current match number. Simultaneous matches („arenas") are numbered starting with 1. All commands specific to a match operate on the current match.

```
> GET_ARENA
< OK
< 5 1
```

SELECT_ARENA

Changes your current arena. Accepts one argument: arena (match) number. This number must be between 1 and maximum number of simultaneous rounds from the constants. Only arenas from 1 to number of matches returned by GET_ARENA are active – you can select the other ones, but you can't do anything in it.

```
> SELECT_ARENA 3
< OK
```

GET_CONSTANTS

Returns a number of constants, all in one line:

- radius of the playfield (which is a circle with center at $(0, 0)$),
- radius of the spaceships,
- radius of the explosions,

- maximum number of commands you can give to a rocket,
- maximum number of simultaneous matches you can play,
- maximum acceleration of a spaceship (length of the vector),
- maximum acceleration of a rocket (length of the vector),
- maximum absolute value of angular acceleration of a rocket,
- maximum starting velocity of a rocket (length of the vector),
- maximum absolute value for starting angular velocity of a rocket,
- number of points you get for destroying an enemy spaceship,
- number of turns you have to wait to launch another rocket (cooldown),
- number of turns in a match.

First five constants won't change for a given server, other can sometimes change between tournaments.

```
> GET_CONSTANTS
< OK
< 500.00000 12.00000 20.00000 100 5 0.05000 0.10000 0.02000 1.50000 0.25000 100 75 3000
```

GET_SHIPS

Returns remaining spaceships from the current match. On the first line it returns the number of ships, the one line for each spaceship. Each ship is described by the following parameters:

- ID number of the spaceship (unique in this match),
- 1 if it is your ship, 0 otherwise,
- X and Y coordinates of the center of the circle representing the spaceship,
- velocity vector components (X and Y),
- acceleration vector components (X and Y),
- number of turns you have to wait before you can launch another rocket. If it's 0 you can already launch it.

```
> GET_SHIPS
< OK
< 6
< 0 1 0.00000 400.00000 0.00000 0.00000 0.00000 0.00000 0
< 1 1 100.00000 400.00000 0.00000 0.00000 0.00000 0.00000 0
< 2 1 -100.00000 400.00000 0.00000 0.00000 0.00000 0.00000 0
< 3 0 0.00000 -400.00000 0.00000 0.00000 0.00000 0.00000 0
< 4 0 100.00000 -400.00000 0.00000 0.00000 0.00000 0.00000 0
< 5 0 -100.00000 -400.00000 0.00000 0.00000 0.00000 0.00000 0
```

GET_ROCKETS

Returns rockets from the current match. On the first line it returns the number of rockets, then one line for each rocket. Each rocket is described by following parameters:

- ID number of the rocket (unique in this match),
- 1 if you launched this rocket, 0 if the enemy did,
- X and Y coordinates of the rocket,
- angle (direction) of the rocket, in radians
- velocity vector components (X and Y),

- angular velocity (positive number means counter-clockwise movement, negative number is clockwise),
- current engine power (like in `SET_ENGINE`),
- current thrusters power (like in `SET_THRUSTERS`).

```
> GET_ROCKETS
< OK
< 3
< 0 1 121.70535 -182.07828 0.09927 0.14867 1.49261 0.12000 0.52132 -1.00000
< 1 1 39.42168 -325.82500 0.48850 0.70396 1.32455 0.00000 0.00000 1.00000
< 2 0 -75.42286 -373.69478 0.75145 1.02405 1.09605 0.00000 0.92220 0.52502
```

GET_EXPLOSIONS

Returns explosions from the current match. On the first line it returns a number of explosions, the one line for each explosion. Each explosions is described by following parameters:

- ID number of the explosions (the are the same as the ID of a rocket that caused the explosion),
- 1 if your rocket caused this explosion, 0 if enemy caused it,
- X and Y coordinates of the center of the circle representing the explosion.

```
> GET_EXPLOSIONS
< OK
< 0 1 121.70535 -182.07828
< 1 1 39.42168 -325.82500
< 2 0 -75.42286 -373.69478
```

GET_ASTEROIDS

Returns remaining asteroids from the current match. On the first line it returns a number of asteroids, then one line for each asteroid. Each asteroid is described by following parameters:

- ID number of the asteroid (unique in this match),
- X and Y coordinates of the center of the circle representing the asteroid,
- radius of the asteroid,
- number of hit points of the asteroid,
- number of points you get for damaging the asteroid.

```
> GET_ASTEROIDS
< OK
< 5
< 12 -64.81761 96.17701 25.64017 7 33
< 13 36.50841 226.56936 9.32771 2 45
< 14 -36.50841 -226.56936 9.32771 2 45
< 15 323.62894 -261.31613 38.70985 13 14
< 16 -323.62894 261.31613 38.70985 13 14
```

GET_STATE

Returns the number of the current turn and you current score in the current match.

```
> GET_STATE
< OK
< 110 55
```

SET_SHIP_ENGINE

Sets an acceleration vector for a given ship in the current match. It takes an ID of the ship, the X component and the Y component of the vector. If the length of this vector is greater than maximum possible acceleration, it is shortened. You can issue this command multiple times in a turn, but only the last one matters.

```
> SET_SHIP_ENGINE 3 5.5 3.71
< OK
```

SET_SHIP_BRAKE

Stops the spaceship (sets its velocity to 0). It takes an ID of the ship. You can issue this command multiple times in a turn, but only the last one matters.

```
> SET_SHIP_BRAKE 5
< OK
```

LAUNCH_ROCKET

Launches a rocket. It takes the following arguments:

- ID of the spaceship that launches the rocket,
- starting velocity vector (X and Y coordinates),
- starting angle of the rocket in radians,
- starting angular velocity of the rocket.

If the length of the velocity vector is greater than maximum starting velocity, it is shortened. Absolute value of starting angular velocity cannot be greater than the maximum absolute value (specified in constants). Positive values denote counter-clockwise direction, negative values mean clockwise direction.

This command is special - after it succeeds with `OK`, you enter a programming mode. Now you can only execute rocket programming commands, until you exit the programming mode by using `END` (which returns `OK`, followed by ID of the rocket you just launched). Rocket programming commands don't return anything if they succeed (not even `OK`), but they do return errors. You can't exceed commands limit specified in the constants. Here is an example:

```
> LAUNCH_ROCKET 3 1.0 -0.2 1.57 -0.25
< OK
> DELAY 5
> SET_ENGINE 1.0
> EXPLODE
> END
< OK
< 10
```

Rocket commands

These commands only work in the programming mode, and they don't return anything when they succeed (apart from `END`).

SET_ENGINE

Takes a real number **r** between 0 and 1 inclusive. It sets the length of the rocket acceleration vector to **r***(maximum acceleration of a rocket) (so 0 means engine turned off, 1 means full power).

```
> SET_ENGINE 0.77
```

SET_THRUSTERS

Takes a real number **r** between -1 and 1 inclusive. It sets the angular acceleration vector to **r***(maximum absolute value of angular acceleration). As always, positive numbers mean counter-clockwise direction, and negative numbers mean clockwise.

```
> SET_THRUSTERS -0.11
```

EXPLODE

Tells the rocket to explode.

```
> EXPLODE
```

DELAY

Takes an integer **x**. This command doesn't do anything, but it takes **x** turns to execute. **x** can't be greater than the number of turns in a match, and it has to be positive.

```
> DELAY 5
```

END

Exits the programming mode, and finishes the sequence of commands for the rocket to execute. This is the only way to exit programming mode. Returns OK, and an ID a rocket that you just launched.

```
> END
< OK
< 10
```

Error codes

- 201 Command allowed only in programing mode
- 202 Command not allowed in programing mode
- 203 Provided NaN number
- 204 Provided Inf number
- 301 Wrong Space Ship Id
- 302 Space Ship does not belong to player
- 303 Angular velocity is greater than maximum limit
- 304 Space Ship rocket launch is not ready yet
- 305 Value must be between 1 and <number of turns per round>
- 306 Rocket commands limit exceeded
- 307 Value must be between -1 and 1
- 308 Value must be between 0 and 1
- 309 Wrong arena number
- 310 Selected arena slot is inactive