Asteroid Robots

About this task

Thank you for taking the time to apply! As part of the process we'd like to to complete this take-home task. Hopefully it should not take very long, and certainly not more than 2 hours.

Please treat this task as though it is a real project and do everything just as you normally would. Specifically, please try to do Test-Driven Design.

About the robots

The European Space Agency (ESA) is planning to send some robots to an asteroid.

The asteroid is curiously rectangular.

A robot position is represented by a pair of co-ordinates and the robot's current bearing. An example position might be (1, 3, South) which means that the robot is 1 mile east and three miles north and is facing south.

The co-ordinate of the position one place North from (0,0) is (0, 1).

In order to control a rover, ESA sends a series of JSON messages, each one on a new line. The allowed messages are:

1. A message stating the size of the asteroid. This will always be the first message.
2. A message stating the position of a new robot.
3. A message telling the current robot to move.

A message stating the size of an asteroid looks like:

```
{"type": "asteroid": "size": {"x": 5, "y": 5}
```

A new robot message looks like:

```
{"type": "new-robot", "position": {"x": 0, "y": 1}, "bearing":
"north"}
```

A movement message looks like:

```
{"type": "move", "movement": "turn-left"} # or "turn-right" or
"move-forward"
```

The output should be a series of JSON messages, each one on a new line, describing the position of the robots after all of the input has been executed. This message should look like:

```
{"type": "robot": , "position": {"x": 7, "y": 3}, "bearing":
"south"}
```

Your program

Please make your program as easy to build and run as possible and if necessary include clear instructions.

Your program should receive messages from a text file passed as an argument and should output messages on stdout. For example:

```
robots.py instructions.txt
```

Do not include your name in anywhere in any of your source files or in your instructions.

Worked example

Input

```
{"type": "asteroid", "size": {"x": 5, "y": 5}}
{"type": "new-robot", "position": {"x": 1, "y": 2}, "bearing": "north"}
{"type": "move", "movement": "turn-left"}
{"type": "move", "movement": "move-forward"}
{"type": "move", "movement": "turn-left"}
{"type": "move", "movement": "move-forward"}
{"type": "move", "movement": "turn-left"}
{"type": "move", "movement": "move-forward"}
{"type": "move", "movement": "turn-left"}
{"type": "move", "movement": "move-forward"}
{"type": "move", "movement": "move-forward"}
{"type": "new-robot", "position": {"x": 3, "y": 3}, "bearing": "east"}
{"type": "move", "movement": "move-forward"}
{"type": "move", "movement": "move-forward"}
{"type": "move", "movement": "turn-right"}
{"type": "move", "movement": "move-forward"}
{"type": "move", "movement": "move-forward"}
{"type": "move", "movement": "turn-right"}
{"type": "move", "movement": "move-forward"}
{"type": "move", "movement": "turn-right"}
{"type": "move", "movement": "turn-right"}
{"type": "move", "movement": "move-forward"}
```

Output

```
{"type": "robot": , "position": {"x": 1, "y": 3}, "bearing": "north"}
{"type": "robot": , "position": {"x": 5, "y": 1}, "bearing": "east"}
```