**Asteroid Robots**

**Introduction**

HI! Thanks for applying to join Black Cow Technology. As part of the process we'd like you to complete this short take-home task. It shouldn't take very long, certainly not more than a *few hours*.

Please treat this task as though it were a real project and do everything just as you normally would (**think source control**).

Additionally, please *try* to apply the principles of [test-driven development](https://en.wikipedia.org/wiki/Test-driven_development) using [pytest](https://docs.pytest.org/en/latest/).

**Show Me The Robots!**

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

The surface of the asteroid is mapped to a rectangular grid.

A robot's ***position*** on an asteroid is represented by a pair of co-ordinates and its *current* bearing (behavior) .An example position might be (1, 3, South) which means that *the robot* is one mile East and three miles North of the asteroid's origin and *is facing* South.

The co-ordinate of the position one place North of the asteroid's origin (0, 0) is (0, 1).

In order to control **a robot**, the ESA sends a *series* *JSON messages*, *each* one on a new line.

The allowed message *types* are:

* A message *stating* *the size* of the asteroid - this will **always** be ***the first*** message
* A message *stating* the position of **a *new*** robot
* 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*** (VP: *existing robot*) message looks like:

{"type": "move", "movement": "turn-left"}

The other allowed values for "movement" are "turn-right" and "move-forward".

VP: (Q) how do we know to whom is the message???

**The Program**

Your program *should consume* *a series* of **JSON** commands provided in a text file, one line per command.

Your program *should output* a series of **JSON** messages, each on a new line, describing ***the final positions of the robots*** *after* all of the input commands have been executed.

*An output message* should look like this, with **one** message ***per robot***:

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

Please make your program as easy to 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:

$ python robots.py instructions.txt

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

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

***Consideration*** should be given to *the* ***amount of memory*** used by the program. Additionally think about, and explain if necessary, how the program *treats* ***the boundaries*** of **the grid** representing the asteroid.

**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"}