8.5 Assignment 5, due Thursday 2018-03-29, at 11:00pm

For this assignment, you get to simulate a 3-dimensional universe full of ants.

8.5.1 Your task

You are to implement a program ants, which takes from stdin a sequence of commands to an army of 256 ants, each labeled permanently by one of the 256 possible values of a char. The ants move around in a 232 × 232 × 232 3-dimensional universe with coordinates given by ints. Initially the universe is entirely filled with space characters (' ' == 0x20), but over time the ants may overwrite these with other values. Every ant starts at position (0, 0, 0). It is permitted for multiple ants to be in the same position.

At any time, there is exactly one ant receiving commands, which is referred to as the *current ant*.  
This ant receives a sequence of commands from stdin, each consisting of a single character, the effects of which are described in the list of commands below.

The initial current ant is given by whatever character appears first on stdin.  
Subsequently, which ant is the current ant may be set by the \n command.

8.5.2 List of commands and their effects

**h**

Move the current ant one position west.

**j**

Move the current ant one position south.

**k**

Move the current ant one position north.

**l**

Move the current ant one position east.

**'<'**

Move the current ant one position up.

**'>'**

Move the current ant one position down.

**\***

Jump the current ant from its current position (*x*, *y*, *z*) to position (2*x*, 2*y*, 2*z*).

**.**

Write the current ant's character to its current position.

**?**

Read the character at the current ant's current position and print it to stdout.

**\n*character***

Set the current ant to *character*. If \n is immediately followed by end-of-file, ignore it and exit.

**EOF**

Exit.

Any other characters are ignored, which allows for comments written in all caps that avoid certain punctuation characters.

An ant that attempts to walk off an edge of the plane or jump to a coordinate outside the range of an int wraps around using the usual C int overflow rules (in other words, you don't need to test for overflow).

8.5.3 Examples

#?

makes ant # print a space.

#lll.

-lll?l?h?

makes ant # move 3 positions east and write #. Ant - then moves 3 positions east, prints the #, moves another position east, prints a space, moves a position west, and prints the # again.

h.?

e.?

lj.??

oh.?

,.?k?

w.?

o?

r.?

l?

d.?

.?

prints hello, world followed by a newline. (Note the use of "newline ant" in the last command.)

#j.\*.\*.\*.k.

=j?j?j?j?j?j?j?j?j?j?

moves ant # south 7 positions using a combinations of moves and doublings, then sends ant = after it to show all the places ant # landed.

#.hj<lk>?

moves ant # around a 3-dimensional path that ends back where it started.