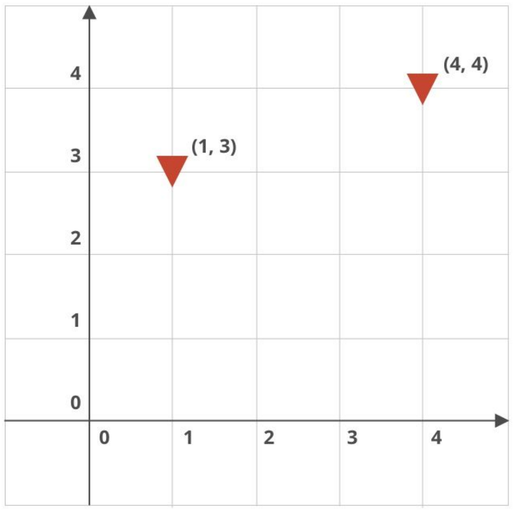
Challenge: Pizzabot

As part of our continuing commitment to the latest cutting-edge pizza technology research, Slice is working on a robot that delivers pizza. We call it (dramatic pause): Pizzabot. Your task is to instruct Pizzabot on how to deliver pizzas to all the houses in a neighborhood.
In more specific terms, given a grid (where each point on the grid is one house) and a list of points representing houses in need of pizza delivery, return a list of instructions for getting Pizzabot to those locations and delivering. An instruction is one of:

N: Move north
S: Move south
E: Move east
W: Move west
D: Drop pizza

Pizzabot always starts at the origin point, (0, 0). As with a Cartesian plane, this point lies at the most south-westerly point of the grid. See below for an example grid :



Therefore, given the following input string:

```
5x5 (1, 3) (4, 4)
```

one correct solution would be:

```
ENNNDEEEND
```

In other words: move east once and north thrice; drop a pizza; move east thrice and north once; drop a final pizza.

Your solution should have a `` `pizzabot` `` command, executable from the command line, i.e.

```
./pizzabot "5x5 (1, 3) (4, 4)"
```

There are multiple correct ways to navigate between locations. We do not take optimality of route into account when grading: all correct solutions are good solutions.

To complete the challenge, please solve for the following exact input string:
5x5 (0, 0) (1, 3) (4, 4) (4, 2) (4, 2) (0, 1) (3, 2) (2, 3) (4, 1)

Keep it simple, and have fun!

Additional for Kotlin and Swift submissions
If you'd prefer to avoid stdin, or work predominantly in a platform that makes it difficult to use, the equivalent solution expressed as an integration test is just fine. The API is entirely up to you, as long as the test accepts and returns properly formatted strings, e.g.

```
assertEqual(pizzabot("5x5 (1, 3) (4, 4)"), "ENNNDEEEND")
```

Notes

1) Please submit the solution to your challenge as a tarball, with clear instructions on how to execute it.
2) When the test is anonymously reviewed by a panel of your potential peers, here's what we're looking for :-

- Correctness:
    - Does the code fulfill all the requirements of the challenge?
- Production Readiness:
    - Is the code well-structured by the standards of the host language?
    - Is the solution maintainable and easy to make changes to?
    - Is the code clean, readable and easy for other team members to understand?
    - Is there appropriate test coverage?
- Fit and polish:

- Is there a README? A build script?
- Are there spelling errors or extraneous comments?
- How does it handle unspecified input?