VivaCity Coding Challenge

How to test your solutions
The challenge uses input/output file based testing. Your solution to each problem needs to read an input from a text file, then output the answer to another text file in the same location with the same name and the suffix ".answer" appended to the name.

Your programs must be designed to take in a single command-line argument that will be an absolute path to the input file. For example, if you use Python , we will run your programs as follows.

```
python your-hash-solution.py /home/johndoe/challenge/input/hash-1.txt
```

In this case, your program should output the answer to /home/johndoe/challenge/input/hash-1.txt.answer.

There are example input and output files provided for each problem, which you can use as a baseline for testing your code. For example, your solution to Problem 1 should be able to read the contents of hash-1.txt in the input folder and produce a file called hash-1.txt.answer in the same folder, whose content is exactly equal to that of the hash-1.txt.example.answer file.

Your code will be tested automatically (using the supplied tests and others) and reviewed manually. Please do not assume that input files will be well- formed (e.g. your code should appropriately handle unexpected characters in the input file). Please use the linux-style newlines in your files (\n, ASCII code 10). Note that your output files should not have a newline at the end of the file.

What we're looking for
Please approach each problem as though you were starting the code for a new VivaCity product. At a high level, a great solution is one which:

- Delivers all of the desired functionality
- Could be picked up by a colleague and understood quickly
- Could be extended or reworked in the future
- Uses the features of your chosen programming language well
- Handles unexpected inputs appropriately

To be specific, your code will be evaluated for:

- Functionality (whether it solves the problem)
- Approach (e.g. algorithm design)
- Code quality (e.g. appropriate use of functions)
- Cleanliness (e.g. variable naming)

- Performance (speed, less important)

What to submit and how
We have sent you this challenge as an archive. Please add your solutions to the 2 problems as separate files at the root of this, as well as a file called **notes.txt**, and then email the zip archive back to us. Feel free to include other files, e.g. tests, util classes, etc.

The notes.txt file should include:
- Brief instructions on how to run your code
- Any notes we should bear in mind (e.g. why you made an unusual choice)
- A brief note on what you found hardest about the challenge
- A brief note on how you could improve your code further
- Any feedback to us on how to improve either the challenge itself or the recruitment process so far

We review all submissions anonymously to remove any chance of bias. Please do not include your name or any personally identifiable information in any files in your submission.

We realize some editors and package managers do this automatically, so please check all the files you are submitting and make sure that your name has not been added.

Problem: Pathfinding

Problem description

This problem is to create a map and then find the shortest path from a start location to an end location. Think of it as being a pirate in search of treasure.

The input file, a coded message left behind by a fellow pirate, contains comma-separated coordinates, which detail your starting location, the location of the treasure, and the locations of dangerous reefs on the ocean, that you cannot sail over. The first correctly formatted coordinate in a file is your starting position, and the last correctly formatted coordinate is the location of the treasure. All other coordinates are reefs that you cannot sail through. The map always starts at (0,0) and its extent is defined by reefs only.

Before you can find the shortest path to the treasure, you need to process the information you have collected from your pirate buddies while sailing the seas.

Start by developing a parsing function which takes the input file, and produces a map.

Each coordinate file contains data in the following form: x4y1,x1y2,x5y2. The first number is the x coordinate, the second is the y coordinate. Exclude any coordinates that are not in the specified format.

Once you have created the map, you'll have to find the shortest way to get to the treasure. Use a path-finding algorithm to find the steps that take you from the start position to the end location along the shortest route. You can only sail directly up, down, left or right (no diagonal sailing), and you are not allowed to sail outside of the map.

Once you have found the shortest path, output the complete map to a file using the following legend:

Sea: .
Reef: x
Start: S
End (treasure): E
Path to take: O
If you find that there is no path in the ocean, or that your start or end position doesn't make sense (e.g. it lies outside the map), output the word "error" to the output file. Make sure your program exits gracefully instead of throwing an exception in these cases.

<u>Example</u>
Input:

```
x0y0,x0y1,x1y1,
x3y2,x2y2
```

The start coordinate is hence x0y0, while the treasure lies at x2y2. We mark these as S and E. That gives us a map like this:

```
S...
xx..
..Ex
```

We find the shortest path that connects the start and end points by sailing through the ocean. We mark the path with O, which yields us the final map. Output file:

```
SOO.
xxO.
..Ex
```

<u>Extra notes:</u>
For this problem, please find a suitable library to use for pathfinding, rather than implementing your own algorithm
If there are multiple shortest paths, output just one of them
The oceans are vast. Coordinates may span multiple lines and maps can be quite large
Coordinates are zero-indexed
The origin of the coordinate axes is at the top left

Remember
Your programs must be designed to take in a single command-line argument that will be an absolute path to the input file. For example, if you use Python , we will run your programs as follows.

```
python your-pathfinding-solution.py /home/johndoe/challenge/input/path-1.txt
```

And again, please do not add your name or other personally identifiable information to the solutions you are submitting - this helps us keep reviews anonymous and unbiased.

Good Luck!