Coding Test

Introduction

The DI coding test is designed to be simple enough that anyone applying should be able to provide a working solution. As such we hope you have the time to spend effort on providing a high-quality example of your ability to write code and engineer software.

Problem definition

You will write a program that takes a single command-line argument that species a file path which points to a file specifying some *source text* and a *search term*. Your job is to search the source text for matches of the search term, and output all the matches as defined below.

Input file

The input file is a contiguous list of lines, the final line specifies the *search term,* all other lines specifying *source text:*

```
<source_text>
<source_text>
<source_text>
<search_term>
```

Examples of the input file can be find in the ***Example*** section below.

Source text

The *source text* consists of lines of strings, with each line containing three words embedded in symbols, numbers and spaces:

For example the line

```
this is one
```

contains the words: `this` , `is` and `one` . And the line:

```
908^)-234 923this-++-23is./<.";][}"another-=&^5
```

contains the words: `this` , `is` and `another`

Search term

The *search term* is always on the last line of the file, and contains a single word.

Match

A line from the source text is considered a match if any word on the line contains the search term as a substring.

For example the line

```
this is one
```

would match the search terms `this`, `is`, `one`, `his`, `on`, `s` amongst many others. But would not match the search terms `siht` , `b` , `iso`, `thisisone` amongst many, many others.

Output

You are expected to output the words on a matching line formatted as single space separate square-bracket-enclosed lists. `[this is ok]` is correct, but all of:

- `[this, no, good]`
- `{this no good}`
- `[this,no,good]`
- `[this no good]` (multiple spaces)

are not correctly formatted.

If there are multiple lines that match, you should output them in the order they appear in the source text.

Examples

The following examples provide some simple test cases to get started with.

Example1

Given the input file:

```
cat sees me
mary likes trees
up the hill
ee
```

Your program should output:

```
[cat sees me]
[mary likes trees]
```

Example2

Given the input file:

```
"Alice was beginning...
to_get9_!very
1111tired1111of1111sitting1111
     by her_sister.
on9the bank,
and""of""having
nothing to do!!!
er
```

Your program should output:

```
[to get very]
[by her sister]
```

Solution delivery

We will run automated testing of your solution, and so we hope to be able to set it up and run it easily. Please make sure there are clear instructions about how to compile/ build/ install your solution so that we can run the command:

```
solution <path_to_file>
```

to run your solution on the provided file.

You may want to add a description about your solution, as well as describing any choices and assumptions you had to make. Please provide this in a `README.md` file. You are free to make any assumptions you feel necessary, please provide a description of them along with a justification in the `README.md`

You are free to choose any programming language you wish.