Exercises: String Processing

This document defines the exercises for "Java Advanced" course @ Software University. Please submit your solutions (source code) of all below described problems in Judge.

1. Count Substring Occurrences

Write a program to find **how many times** a given **string appears in a given text** as substring. The text is given at the first input line. The search string is given at the second input line. The output should be an integer number. Ignore the **character casing**. **Overlapping** between occurrences is **allowed**.

Examples

Input	Output
<pre>Welcome to the Software University (SoftUni)! Welcome to programming. Programming is wellness for developers, said Maxwell. wel</pre>	4
aaaaaa aa	5
ababa caba aba	3
Welcome to SoftUni Java	0

Hints

• For ignoring the character casing, you will need two strings, the one uppercase, and the other lowercase.

2. Sum Big Numbers

You are given two lines - each can be a really big number (0 to 10^{50}). You must display the sum of these numbers.

Note: do not use the **BigInteger** or **BigDecimal** classes for solving this problem.

Input	Output	Input	Output
23	46	9999	10000
23		1	

Input	Output
923847238931983192462832102	934573817465075391826664309019448
934572893617836459843471846187346	



















3. Text Filter

Write a program that takes a string of banned words and a text. All words included in the ban list should be replaced with "*", equal to the word's length. The entries in the ban list will be separated with a comma and a space

The ban list should be entered on the first input line and the text on the second input line.

Examples

Input	Output
Linux, Windows It is not Linux, it is GNU/Linux. Linux is merely the kernel, while GNU adds the functionality. Therefore we owe it to them by calling the OS GNU/Linux! Sincerely, a Windows client	It is not *****, it is GNU/****. ***** is merely the kernel, while GNU adds the functionality. Therefore we owe it to them by calling the OS GNU/****! Sincerely, a ****** client

4. Unicode Characters

Write a program that converts a string to a sequence of Unicode character literals.

Examples

Input	Output
Hi!	\u0048\u0069\u0021
What?!?	\u0057\u0068\u0061\u0074\u003f\u0021\u003f
SoftUni	\u0053\u006f\u0066\u0074\u0055\u006e\u0069

Hints

Look here for some of the StringBuilder methods. The Problem is very easy if you the find right method.

5. Palindromes

Write a program that extracts from a given text all palindromes, e.g. ABBA, lamal, exe and prints them separated by a comma and a space. Use spaces, commas, dots, question marks and exclamation marks as word delimiters. Print all unique palindromes, sorted lexicographically.

Examples

Input	Output
Hi,exe? ABBA! Hog fully a string. Bob	[ABBA, a, exe]
aibohphobia is fear of palindromes ahahaha	[ahahaha, aibohphobia]
aSantAAtnaSa is a rare sight	[a, aSantAAtnaSa]

Hints

TreeSet is a data structure, which sorts words lexicographically immediately after adding a new one.

Set<String> palindromes = new TreeSet<>();





















6. Magic Exchangeable Words

Write a method that takes as input two strings, and returns true if they are exchangeable. Exchangeable are words, where the characters in the first string can be replaced to get the second string.

Example: "egg" and "add" are exchangeable, but "aabccbb" and "nnoppzz" are not. (First 'b' corresponds to 'o', but then it also corresponds to 'z').

The two words may not have the same length, if such is the case they are exchangeable only if the longer one doesn't have more types of characters then the shorter one ("Clint" and "Eastwaat" are exchangeable because 'a' and 't' are already mapped as 'I' and 'n', but "Clint" and "Eastwood" aren't exchangeable because 'o' and 'd' are not contained in "Clint").

Examples

Input	Output
gosho hapka	true
aabbaa ddeedd	true
foo bar	false
Clint Eastwood	false

7. * Letters Change Numbers

Nakov likes Math. But he also likes the English alphabet a lot. He invented a game with numbers and letters from the English alphabet. The game was simple. You get a string consisting of a number between two letters. Depending on whether the letter was in front of the number or after it you would perform different mathematical operations on the number to achieve the result.

First you start with the letter before the number. If it's Uppercase you divide the number by the letter's position in the alphabet. If it's lowercase you multiply the number with the letter's position. Then you move to the letter after the number. If it's Uppercase you subtract its position from the resulted number. If it's lowercase you add its position to the resulted number. But the game became too easy for Nakov really quick. He decided to complicate it a bit by doing the same but with multiple strings keeping track of only the total sum of all results. Once he started to solve this with more strings and bigger numbers it became quite hard to do it only in his mind. So he kindly asks you to write a program that calculates the sum of all numbers after the operations on each number have been done.

For example, you are given the sequence "A12b s17G". We have two strings - "A12b" and "s17G". We do the operations on each and sum them. We start with the letter before the number on the first string. A is Uppercase and its position in the alphabet is 1. So we divide the number 12 with the position 1 (12/1 = 12). Then we move to the letter after the number. b is lowercase and its position is 2. So we add 2 to the resulted number (12+2=14). Similarly for the second string s is lowercase and its position is 19 so we multiply it with the number (17*19 = 323). Then we have Uppercase G with position 7, so we subtract it from the resulted number (323 - 7 = 316). Finally we sum the 2 results and we get 14 + 316=330;

Input

The input comes from the console as a single line, holding the sequence of strings. Strings are separated by one or more white spaces.

The input data will always be valid and in the format described. There is no need to check it explicitly.



















Output

Print at the console a single number: the total sum of all processed numbers rounded up to two digits after the decimal separator.

Constraints

- The **count** of the strings will be in the range [1 ... 10].
- The numbers between the letters will be integers in range [1 ... 2 147 483 647].
- Time limit: 0.3 sec. Memory limit: 16 MB.

Examples

Input	Output	Comment
A12b s17G	330.00	12/1=12, 12+2=14, 17*19=323, 323-7=316, 14+316=330
P34562Z q2576f H456z	46015.13	
a1A	0.00	

8. ** Melrah Shake

You are given a string of random characters, and a pattern of random characters. You need to "shake off" (remove) all of the border occurrences of that pattern, in other words, the very first match and the very last match of the pattern you find in the string.

When you successfully shake off a match, you remove from the pattern the character which corresponds to the index equal to the pattern's length / 2. Then you continue to shake off the border occurrences of the new pattern until the pattern becomes empty or until there is less than the - needed for shake, matches in the remaining string.

In case you have found at least **two** matches, and you have successfully shaken them off, you print "Shaked it." on the console. Otherwise you print "No shake.", the remains of the main string, and you end the program. See the examples for more info.

Input

- The input will consist only of two lines.
- On the first line you will get a string of random characters.
- On the second line you will receive the pattern and that ends the input sequence.

Output

- You must print "Shaked it." for every time you successfully do the melrah shake.
- If the melrah shake fails, you print "No shake.", and on the next line you print what has remained of the main string.

Constraints

- The two strings may contain **ANY** ASCII character.
- Allowed time/memory: 250ms/16MB.

Input	Output
-------	--------























astalavista baby	Shaked it.
sta	No shake.
	alavi baby

Input	Output
##mtm!!mm.mm*mtm.#	Shaked it.
mtm	Shaked it.
	No shake.
	##!!.*.#

9. Match Full Name

Write a regular expression to match a valid full name. If you catch a valid match, print it as an output.

A valid full name:

- Consists of two words
- Each word starts with a capital letter
- Each word contains only lowercase letters afterwards
- Each word should be at least two letters long
- The two words should be separated by a single space

Read lines until you get the "end" command.

Examples

	Input	Output
ivan	ivanov	Ivan Ivanov
Ivan	ivanov	
end	-	

Hints

- Open https://regex101.com/ or a similar regex testing site
- Paste the provided test string and start writing your regex:























- Start with first name
- Add "m" to modifiers which specifies that you are testing a multi-line input



Use character classes to match a single capital letter:

```
/ [A-Z]
                                                         / gm 🦱
```

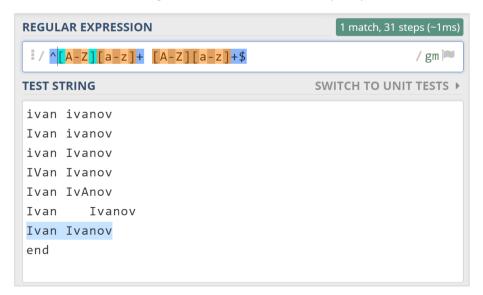
Use character classes and a quantifier to match a series of lowercase letters:

```
/ [A-Z][a-z]+
                                                 / gm
```

Add a single space and repeat the same regex for the second name:

```
[A-Z][a-z]+ [A-Z][a-z]+
                                             / gm
```

Surround the regex with anchors ^ and \$ to specify the start and the end of the regex





















Create your java application using the regex that you've created:

```
String regex = "^[A-Z][a-z]+ [A-Z][a-z]+$";
Scanner sc = new Scanner(System.in);
String text = sc.nextLine();
while (!text.equals("end")) {
    if (Pattern.matches(regex, text)) {
        System.out.println(text);
    text = sc.nextLine();
}
```

Match Phone Number 10.

Write a regular expression to match a valid phone number.

A valid number:

- Starts with "+359"
- Followed by the area code "2"
- Followed by number itself, consisting of 7 digits (separated in two group of 3 and 4 digits respectively)
- Every part of the number should be separated by either a space (' ') or a hyphen ('-'), but not both in a same valid number

Read lines until you get the "end" command.

Input	Output
+359 2 222 2222 +359-2-222-2222 359-2-222-2222 +359/2/222/2222 +359-2 222 2222 +359 2-222-2222 +359-2-222-2222 end	+359 2 222 2222 +359-2-222-2222













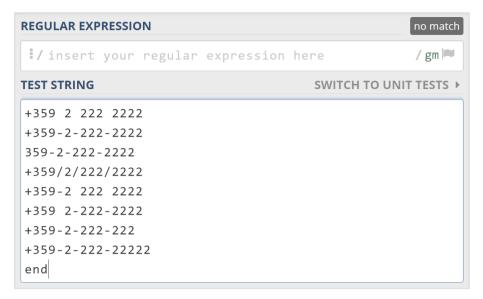








Hints



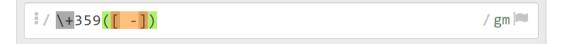
Add "m" to modifiers which specifies that you are testing a multi-line input



Start your regex with the country code, you need to escape the + sign



To match one of the two possible separators, use grouping and a character class



Add the city code which is a literal

```
/ \+359([ -])2
                                                       / gm 🦱
```

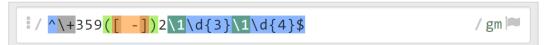
Match the previous separator by using a backreference

```
\+359([ -])2\1
                                                     / gm 🦱
```

Add the next three digits

```
\+359([ -])2\1\d{3}
                                                    / gm 🦱
```

- Do the same for the last separator and the last four digits
- Surround the regular expression with anchors to make sure it wouldn't match longer numbers

























11. Replace <a> Tag

You are given an HTML document given as a string on multiple lines. Write a program that replaces all the tags ... with corresponding tags [URL href=...]...[/URL].

Read lines until you get the "END" command.

Note: a tag can start and end on different lines, but actual keywords like "href=" or the closing tag "" will never be split. For example, you will never get:

ef="http://softuni.bg">SoftUni</

Examples

Input	Output
<pre> SoftUni END</pre>	<pre> (ul> [URL href=http://softuni.bg]SoftUni[/URL] </pre>
 Link END	[URL href="/"] Link[/URL]

Hints

- Use a **StringBuilder** to read the whole input into a single string. Make sure to save the new lines.
- Create a pattern that can match a valid tag on multiple lines.
- Replace all matches.

12. **Extract Emails**

Write a program to extract all email addresses from a given text. The text comes from a single input line. Print the emails on the console, each on a separate line. Emails are considered to be in format **<user>@<host>**, where:

- <user> is a sequence of letters and digits, where '.', '-' and '_' can appear between them.
 - o Valid users: "stephan", "mike03", "s.johnson", "st steward", "softuni-bulgaria", "12345".
 - Invalid users: "--123", ".....", "nakov -", " steve", ".info".
- <host> is a sequence of at least two words, separated by dots '.'. Each word is sequence of letters and can have hyphens '-' between the letters.
 - Valid hosts: "softuni.bg", "software-university.com", "intoprogramming.info", "mail.softuni.org".
 - Invalid hosts: "helloworld", ".unknown.soft.", "invalid-host-", "invalid-".

Emails should start with either a space (' ') or with line start (regex: ^) and end with dot ('.'), comma (','), space (' ') or line end (regex: \$).

- Valid emails: info@softuni-bulgaria.org, kiki@hotmail.co.uk, no-reply@github.com, s.peterson@mail.uu.net, info-bg@software-university.software.academy.
- Invalid emails: --123@gmail.com, ...@mail.bg, .info@info.info, steve@yahoo.cn, mike@helloworld, mike@.unknown.soft., s.johnson@invalid-.

Read lines until you get the "end" command.



















© Software University Foundation (softuni.org). This work is licensed under the CC-BY-NC-SA license.





Examples

Input	Output	
<pre>info@softuni-bulgaria.org,123@gmail.com, kiki@hotmail.co.uk, no-reply@github.com,@mail.bg, s.peterson@mail.uu.net, .info@info.info, info-bg@software- university.software.academy, _steve@yahoo.cn,</pre>	<pre>info@softuni-bulgaria.org kiki@hotmail.co.uk no-reply@github.com s.peterson@mail.uu.net</pre>	
<pre>mike@helloworld, mike@.unknown.soft, s.johnson@invalid- end</pre>	<pre>info-bg@software- university.software.academy</pre>	
Please contact us at: <pre>support@github.com</pre> end	support@github.com	
Just send email to <pre>s.miller@mit.edu</pre> and <pre>j.hopking@york.ac.uk for more information.</pre> end	<pre>s.miller@mit.edu j.hopking@york.ac.uk</pre>	
Many users @ SoftUni confuse email addresses. We @ Softuni.BG provide high-quality training @ home or @ class steve.parker@softuni.de. end	steve.parker@softuni.de	
123@gmail.com,@mail.bg, .info@info.info, _steve@yahoo.cn, mike@helloworld, mike@.unknown.soft., s.johnson@invalid end	(no output)	

Hints

- Learn about positive and negative lookahead and lookbehind.
- Use anchors, character classes, quantifiers and literals

13. Sentence Extractor

Write a program that reads a keyword and some text from the console and prints all sentences from the text, containing that word.

A sentence is any sequence of words that:

ends with a dot ('.'), an exclamation mark ('!') or a question mark ('?').

Input	Output
is	This is my cat!
This is my cat! And this is my dog. We happily live in Paris – the most beautiful city in the world! Isn't it great? Well it is :)	And this is my dog.



















is (no output)
No keyword in this sentence.

14. *Sum of All Values

You are given a **keys string** and a **text string**. Write a program that finds the **start key** and the **end key** from the **keys string** and then **applies them** to the **text string**.

The start key will always stay at the beginning of the keys string. It can contain only letters and underscore and ends just before the first found digit.

The end key will always stay at the end of the keys string. It can contain only letters and underscore and starts just after the last found digit.

Print at the console the sum of all values (only integer and floating-point numbers with dot as a separator are considered valid) in the text string, between a start key and an end key. If the value is 0 then print "The total value is: nothing". If no start key or no end key is found, then print "A key is missing".

Input

The input will be read from the console. The first line will hold the keys string and the second line will hold the text to search.

Output

The output should hold the **result text**, printed in an HTML paragraph.

If the resulting sum is a floating-point number, format it to the second digit after the decimal point and don't print any numbers after the decimal point, if the sum is an integer.

Constraints

- The keys string and text string will hold only ASCII characters (no Unicode).
- Allowed working time: 0.1 seconds. Allowed memory: 16 MB.

Input				
keysString	startKEY12adghfgh243212gadghfgs43endKEY			
textString	textString asdjykulgfjddfsffd startKEY12endKEY qwfrhtyu67543rewghy3tefdgd startKEY123.45endKEY wret34yre startKEY2.6endKEY 213434ytuhrgerweasfd startKEYendKEY startKEY <i>asfdge</i> endKEY			
Output				
The total value is: 138.05				

Input				
keysString	startKEY12a			
textString	asdjykulgfjddfsffd startKEY 12endKEYqwfrhtyu67543rewghy3tefdgdst <mark>a</mark> rtKEY123 .45endKEYwret34yre startKEY 2.6endKEY213434ytuhrgerwe <mark>a</mark> sfd startKEY endKEYst a rtKEYasfdgeendKEY			
Output				





















Input			
keysString	startKEY12		
textString	asdjykulgfjddfsffdstartKEY12endKEYqwfrhtyu67543rewghy3tefdgd		
Output			
A key is missing			

15. *Valid Usernames

You are part of the back-end development team of the next Facebook. You are given a line of usernames, between one of the following symbols: space, "/", "\", "(", ")". First you have to export all valid usernames. A valid username starts with a letter and can contain only letters, digits and "_". It cannot be less than 3 or more than 25 symbols long. Your task is to sum the length of every 2 consecutive valid usernames and print on the console the 2 valid usernames with biggest sum of their lengths, each on a separate line.

Input

The input comes from the console. One line will hold all the data. It will hold **usernames**, divided by the symbols: space, "/", "\", "(", ")".

The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

Print at the console the 2 consecutive valid usernames with the biggest sum of their lengths each on a separate line. If there are 2 or more couples of usernames with the same sum of their lengths, print he left most.

Constraints

- The input line will hold characters in the range [0 ... 9999].
- The usernames should start with a letter and can contain only letters, digits and "".
- The username cannot be less than 3 or more than 25 symbols long.
- Time limit: 0.5 sec. Memory limit: 16 MB.

Input	Output
ds3bhj y1ter/wfsdg 1nh_jgf ds2c_vbg\4htref	wfsdg ds2c_vbg

	Input			Output
min23/ace hahah21 (sasa)	att3454/a/a2/abc	hahah21
				sasa

Input	Output
-------	--------





















chico/ gosho \ sapunerka (3sas) mazut	lelQ_Van4e	mazut
		lelQ_Van4e

**Extract Hyperlinks 16.

Write a program to extract all hyperlinks (<href=...>) from a given text. The text comes from the console on a variable number of lines and ends with the command "END". Print at the console the href values in the text.

The input text is standard HTML code. It may hold many tags and can be formatted in many different forms (with or without whitespace). The <a> elements may have many attributes, not only href. You should extract only the values of the **href** attributes of all **<a>** elements.

Input

The input data comes from the console. It ends when the "END" command is received.

Output

Print at the console the **href** values in the text, each at a separate line, in the order they come from the input.

Constraints

- The input will be well formed HTML fragment (all tags and attributes will be correctly closed).
- Attribute values will never hold tags and hyperlinks, e.g. "<img alt='' />" is invalid.
- Commented links are also extracted.
- The number of input lines will be in the range [1 ... 100].
- Allowed working time: 0.1 seconds. Allowed memory: 16 MB.

Input	Output
<pre> END</pre>	http://softuni.bg
This text has no links END	
html	/
<html></html>	/courses
<head></head>	/forum
<title>Hyperlinks</title>	#
<pre><link href="theme.css" rel="stylesheet"/></pre>	<pre>javascript:alert('hi yo')</pre>
	http://www.nakov.com
<body></body>	#empty
Home<a< td=""><td>#</td></a<>	#
class="selected" href=/courses >Courses	#commented
<a href="</td"><td></td>	
'/forum' >Forum <a <="" class="href" td=""><td></td>	
onclick="go()" href= "#" >Forum	
<a <b="" id="js">href =	



















```
"javascript:alert('hi yo')" class="new">click</a>
<a id='nakov' href =
http://www.nakov.com class='new'>nak</a>
<a href="#empty"></a>
<a id="href">href='fake'<img src='http://abv.bg/i.gif'</pre>
alt='abv'/></a><a href="#">&lt;a href='hello'&gt;</a>
<!-- This code is commented:
 <a href="#commented">commentex hyperlink</a> -->
</body>
END
```

















