# Exercises: Regular Expressions

This document defines the exercises for ["Java Advanced" course @ Software University](https://softuni.bg/courses/java-advanced). Please submit your solutions (source code) of all below described problems in [Judge](https://judge.softuni.bg/).

## 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**

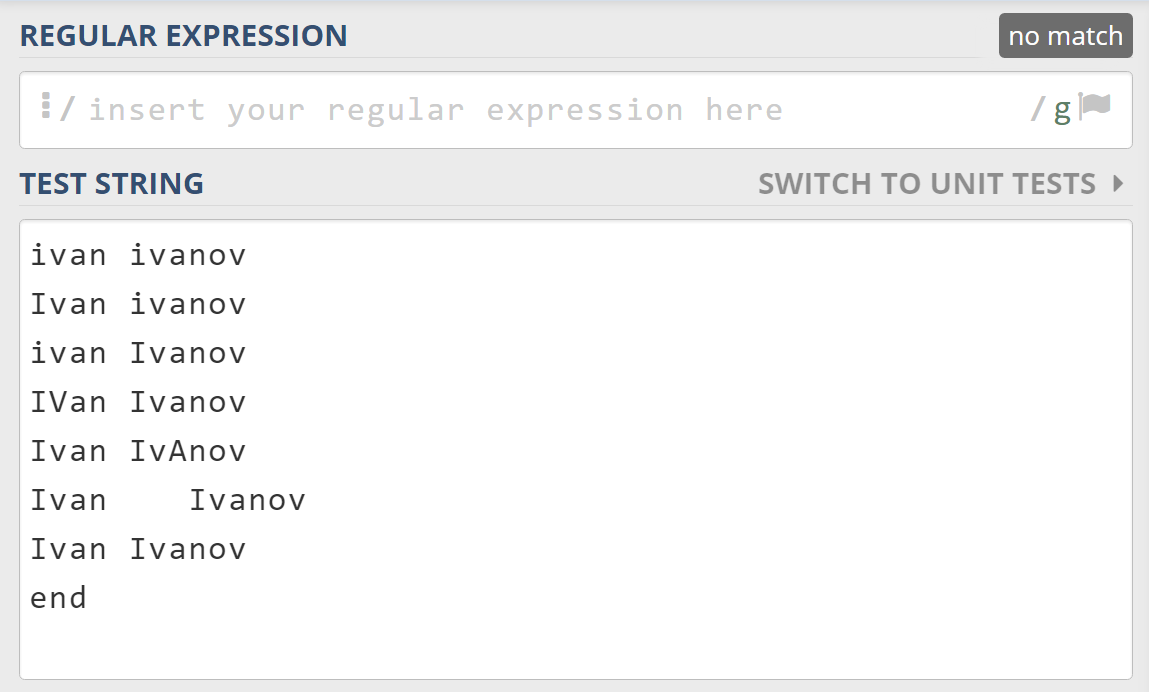
Read lines until you get the "end" command.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| ivan ivanov  Ivan ivanov  ivan Ivanov  IVan Ivanov  Ivan IvAnov  Ivan Ivanov  Ivan Ivanov  end | Ivan Ivanov |

### Hints

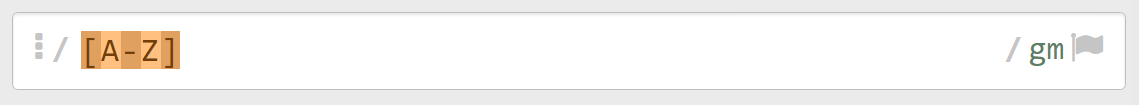
* Open <https://regex101.com/> or a similar regex testing site
* Paste the provided test string and start writing your regex:



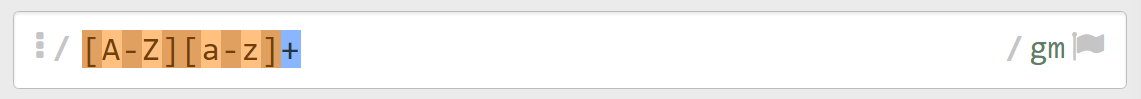
* Start with first name
* Add "m" to modifiers which specifies that you are testing a multi-line input



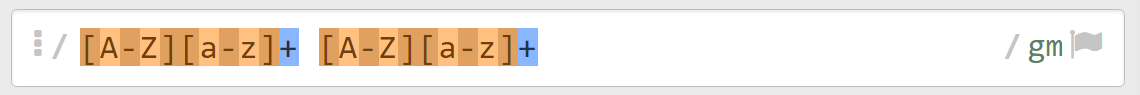
* Use character classes to match a single capital letter:



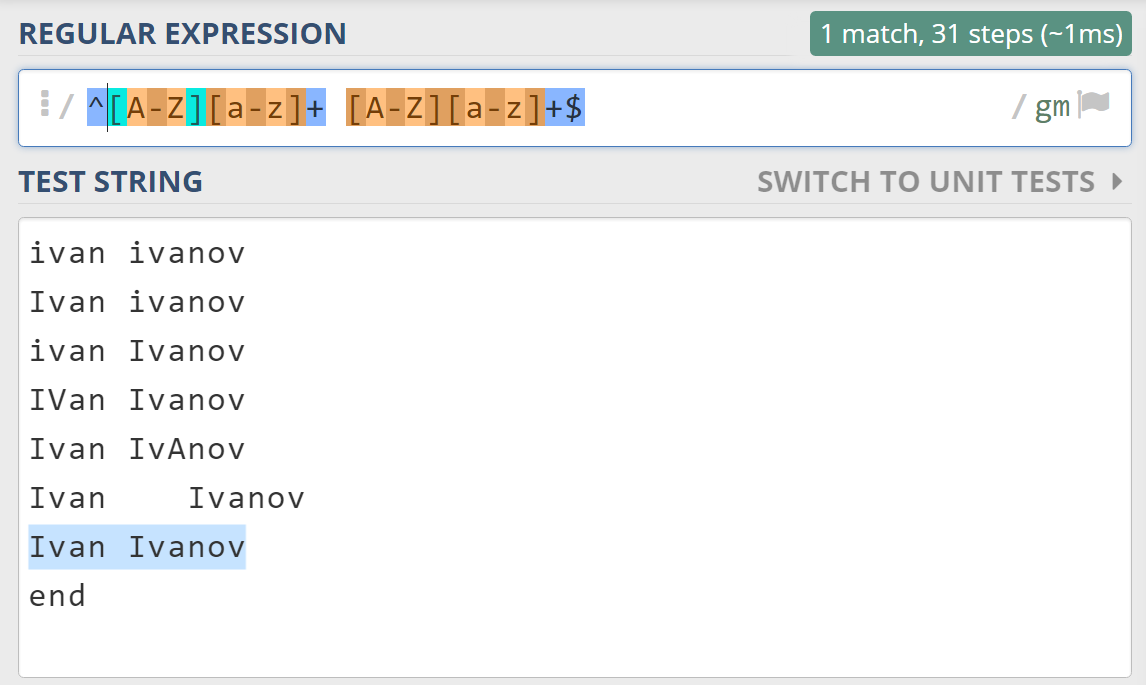
* Use character classes and a quantifier to match a series of lowercase letters:



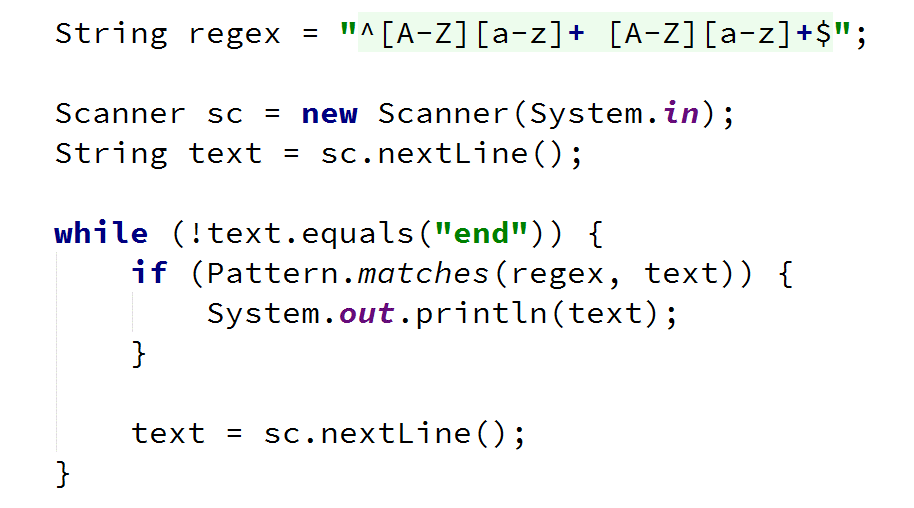
* Add a single space and repeat the same regex for the second name:



* 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:



## Match Phone Number

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

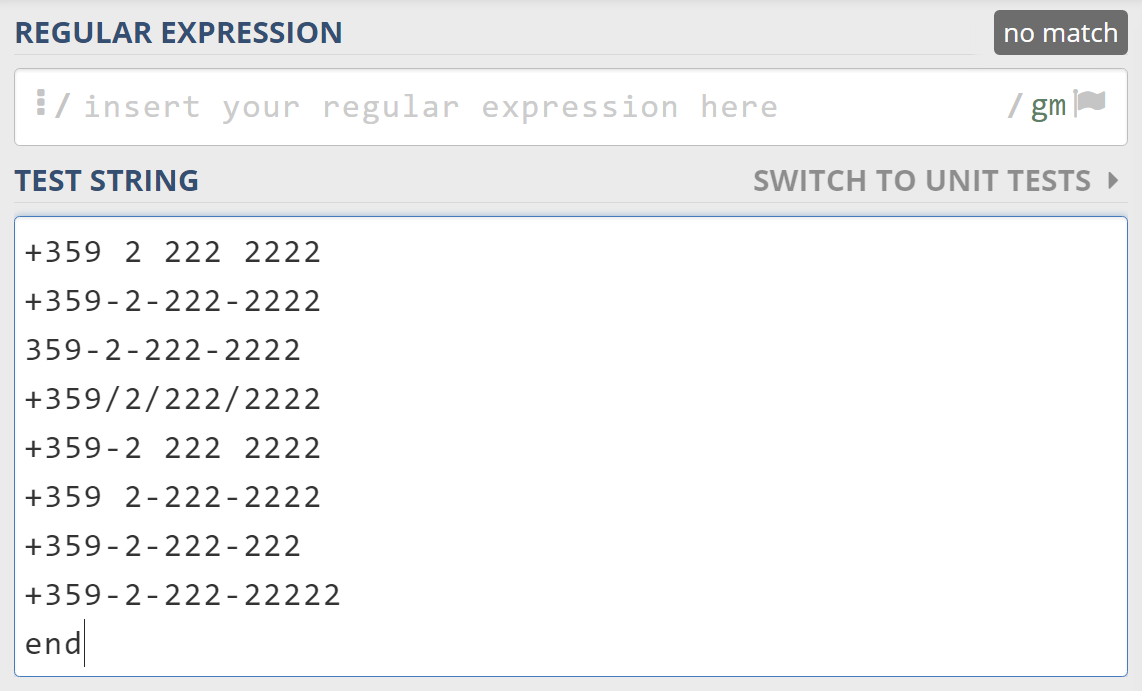
Read lines until you get the **"end"** command.

### Examples

|  |  |
| --- | --- |
| **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  +359-2-222-222  +359-2-222-22222  end | +359 2 222 2222  +359-2-222-2222 |

### Hints

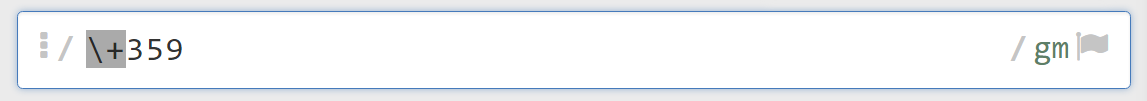
* Open <https://regex101.com/> or a similar regex testing site
* Paste the provided test string and start writing your regex:



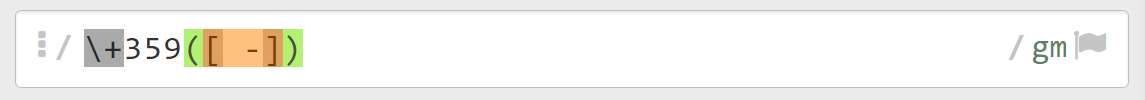
* Add "m" to modifiers which specifies that you are testing a multi-line input



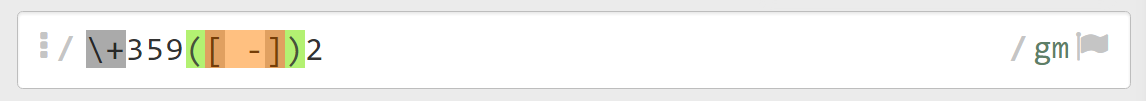
* Start your regex with the country code, you need to escape the + sign



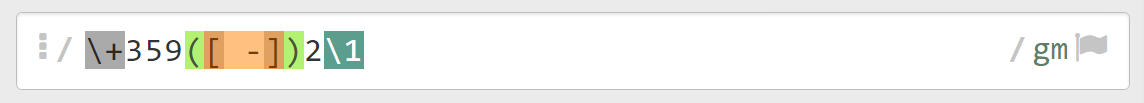
* To match one of the two possible separators, use grouping and a character class



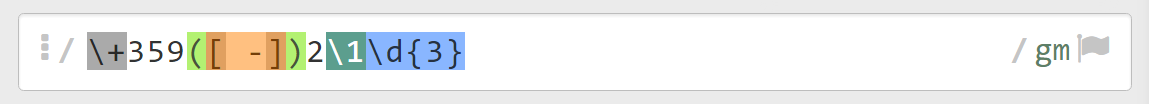
* Add the city code which is a literal



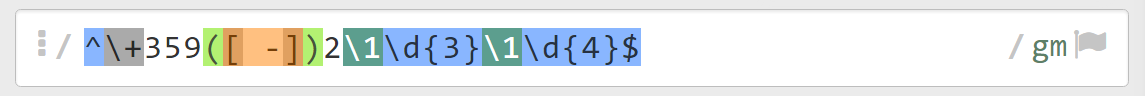
* Match the previous separator by using a backreference



* Add the next three digits



* 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



## Series of Letters

**Read a string** from the console and **replace** all series of **consecutive identical letters** with a **single one**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| aabb | ab |
| abc | abc |
| aaaaabbbbbcdddeeeedssaa | abcdedsa |
| Haaaveee a niceeee Daay! | Have a nice Day! |

### Hints

Use a quantifier for one or more repetitions +, grouping () and backreference constructs \1

## Replace <a> Tag

You are given an HTML document given as a string on multiple lines.

Write a programthat replaces **all the tags** <a href=…>…</a> with corresponding **tags** [URL href=…]…[/URL].

**Note:** a tag can start and end on different lines.

Read lines until you get the **"END"** command.

### Constraints

The href= and </a> elements will never be split on different lines.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| <ul> <li> <a href="http://softuni.bg">SoftUni</a>  </li> </ul>  END | <ul> <li>  **[URL** href=http://softuni.bg**]**SoftUni**[/URL]**  </li>  </ul> |
| <a href="/">  Link</a>  END | **[URL** href="/"**]**  Link**[URL]** |

### Hints

* Use a string builder 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. Note that href= and </a> will never be split.
* Use a string builder to replace all matches.

## 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.
  + **Valid users**: "stephan", "mike03", "s.johnson", "st\_steward", "softuni-bulgaria", "12345".
  + I**nvalid 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.
  + V**alid hosts**: "softuni.bg", "software-university.com", "intoprogramming.info", "mail.softuni.org".
  + I**nvalid 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](mailto:.info@info.info), [\_steve@yahoo.cn](mailto:_steve@yahoo.cn), mike@helloworld, [mike@.unknown.soft](mailto:mike@.unknown.soft)., [s.johnson@invalid-](mailto:s.johnson@invalid-).

Read lines until you get the **"end"** command.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 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, mike@helloworld, mike@.unknown.soft, s.johnson@invalid-  end | info@softuni-bulgaria.org  kiki@hotmail.co.uk  no-reply@github.com  s.peterson@mail.uu.net  info-bg@software-university.software.academy |
| Please contact us at: support@github.com.  end | support@github.com |
| Just send email to s.miller@mit.edu and j.hopking@york.ac.uk for more information.  end | s.miller@mit.edu  j.hopking@york.ac.uk |
| 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](http://www.regular-expressions.info/lookaround.html).
* Use anchors, character classes, quantifiers and literals

## 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:

* ending with dot ('.'), exclamation mark ('!') or question mark ('?').

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| is  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 :) | This is my cat!  And this is my dog. |
| is  No keyword in this sentence. | *(no output)* |

## \*Phone Numbers

You are given a string, holding ASCII characters. Find all **name -> phone number** pairs, format them and print them in an **ordered list** as list items.

The **name** should be at least **one letter long, can contain only letters** and should **always start with an uppercase letter**.

The **phone** **number** should be at least **two digits long**, **should start and end with a digit** (**optionally**, there might be a **“+” in front)** and might **contain** the following symbols in it: **“(“, “)”, “/”, “.”, “-”, “ “** (left and right bracket, slash, dot, dash and whitespace).

Between a name and the phone number there might be **any number of symbols, other than letters and “+”**.

Between the name -> phone number pairs there might be **any number of ASCII symbols**.

The output format should be **<b>[name]:</b> [phone number]** (there is **one** **space between** the name and the phone number). Clear any characters other than digits and “+” from the number when printing the output.

### Input

The input will be read from the console. It will consist of several lines holding the input string. The command "**END**" denotes the end of input.

### Output

The output should hold the **resulting ordered list (on a single line)**, or a single **paragraph,** holding“**No matches!**”

### Constraints

* The **numbers string** will hold only **ASCII** characters (no Unicode).
* Allowed working time: 0.1 seconds. Allowed memory: 16 MB.

### Examples

|  |
| --- |
| **Input** |
| **Angel**$(\*^#**029661234**!@#**Pesho** ,.' **+3592/9653241**;'“:{},. **Ivan** **0888 123 456** **John**-=\_**555.123.4567** Stoian!@#$#@ Gosho )=\_\* **Steven** #$(\*&**+1-(800)-555-2468**  END |
| **Output (li items are separated on new lines for convenience)** |
| <ol><li><b>Angel:</b> 029661234</li>  <li><b>Pesho:</b> +35929653241</li> <li><b>Ivan:</b> 0888123456</li> <li><b>John:</b> 5551234567</li> <li><b>Steven:</b> +18005552468</li></ol> |

|  |
| --- |
| **Input** |
| There are no phone numbers here!!!  END |
| **Output** |
| <p>No matches!</p> |

## \*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. The actual value of the sum should be ***italic.***

### Constraints

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

### Examples

|  |  |
| --- | --- |
| **Input** | |
| keysString | **startKEY**12adghfgh243212gadghfgs43**endKEY** |
| textString | asdjykulgfjddfsffd**startKEY***12***endKEY**qwfrhtyu67543rewghy3tefdgd **startKEY***123.45***endKEY**wret34yre**startKEY***2.6***endKEY**213434ytuhrgerweasfd **startKEYendKEYstartKEY***asfdge***endKEY** |
| **Output** | |
| <p>The total value is: <em>138.05</em></p> | |

|  |  |
| --- | --- |
| **Input** | |
| keysString | **startKEY**12**a** |
| textString | asdjykulgfjddfsffd**startKEY***12endKEYqwfrhtyu67543rewghy3tefdgdst***a**rtKEY123.45endKEYwret34yre**startKEY***2.6endKEY213434ytuhrgerwe***a**sfd **startKEY***endKEYst***a**rtKEYasfdgeendKEY |
| **Output** | |
| <p>The total value is: <em>nothing</em></p> | |

|  |  |
| --- | --- |
| **Input** | |
| keysString | **startKEY**12 |
| textString | asdjykulgfjddfsffdstartKEY12endKEYqwfrhtyu67543rewghy3tefdgd |
| **Output** | |
| <p>A key is missing</p> | |

## \*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.

### Examples

|  |  |
| --- | --- |
| **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 |

## \*Query Mess

**Ivancho** participates in a team **project** with colleagues at **SoftUni**. They have to develop **an application**, but something *mysterious* happened – at the last moment all team members… disappeared! And guess what? He is left **alone** to finish the project. All that is left to do is to parse the input data and store it in a special way, but Ivancho has no idea how to do that! Can you help him?

### Input

The input comes from the console on a variable number of lines and ends when the keyword **"END"** is received.

For each row of the input, the query string contains pairs field=value. Within each pair, the field name and value are separated by an equals sign, '='. The series of pairs are separated by an ampersand, '&'. The question mark is used as a separator and is not part of the query string. A URL query string may contain another URL as value. The input data will always be valid and in the format described. There is no need to check it explicitly.

### Output

**For each input line, print** on the console a line containing the **processed string as follows**:

* key=[value]nextkey=[another value] … etc.

**Multiple whitespace** characters should be reduced to one inside value/key names, but there shouldn’t be any whitespaces before/after extracted **keys** and **values**. If a key **already exists**, the value is added with comma and space after other values of the existing key in the current string. See the **examples** below.

### Constraints

* SPACE is encoded as '+' or "%20". Letters (A-Z and a-z), numbers (0-9), the characters '\*', '-', '.', '\_' and *other non-special symbols* are left as-is.
* Allowed working time: 0.1 seconds. Allowed memory: 16 MB.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| login=student&password=student  END | login=[student]password=[student] |
| **Input** | |
| field=value1&field=value2&field=value3  <http://example.com/over/there?name=ferret>  END | |
| **Output** | |
| field=[value1, value2, value3]  name=[ferret] | |
| **Input** | |
| foo=%20foo&value=+val&foo+=5+%20+203  foo=poo%20&value=valley&dog=wow+  url=https://softuni.bg/trainings/coursesinstances/details/1070  <https://softuni.bg/trainings.asp?trainer=nakov&course=oop&course=php>  END | |
| **Output** | |
| foo=[foo, 5 203]value=[val]  foo=[poo]value=[valley]dog=[wow]  url=[https://softuni.bg/trainings/coursesinstances/details/1070]  trainer=[nakov]course=[oop, php] | |

## \*\*Extract Hyperlinks

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='<a href="hello">' />" 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**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| <a href="http://softuni.bg" class="new"></a>  END | http://softuni.bg |
| <p>This text has no links</p>  END |  |
| <!DOCTYPE html>  <html>  <head>  <title>Hyperlinks</title>  <link href="theme.css" rel="stylesheet" />  </head>  <body>  <ul><li><a **href="/"** id="home">Home</a></li><li><a  class="selected" **href=/courses**>Courses</a>  </li><li><a **href =**  **'/forum'** >Forum</a></li><li><a class="href"  onclick="go()" **href= "#"**>Forum</a></li>  <li><a id="js" **href =**  **"javascript:alert('hi yo')"** class="new">click</a></li>  <li><a id='nakov' **href =**  **http://www.nakov.com** class='new'>nak</a></li></ul>  <a **href="#empty"**></a>  <a id="href">href='fake'<img src='http://abv.bg/i.gif'  alt='abv'/></a><a **href="#"**>&lt;a href='hello'&gt;</a>  <!-- This code is commented:  <a href="#commented">commentex hyperlink</a> -->  </body>  END | /  /courses  /forum  #  javascript:alert('hi yo')  http://www.nakov.com  #empty  #  #commented |

## \*\*Semantic HTML

You are given an **HTML code**, written in the old **non-semantic** style using tags like <div id="header">, <div class="section">, etc. Your task is to write a program that **converts this HTML to semantic HTML** by changing tags like <div id="header"> to their semantic equivalent like <header>.

The non-semantic tags that should be converted are **always** <div>s and have either id or class with one of the following values: "main", "header", "nav", "article", "section", "aside" or "footer". Their corresponding closing tags are always followed by a comment like <!-- header -->, <!-- nav -->, etc. staying at the same line, after the tag.

### Input

The input will be read from the console. It will contain a variable number of lines and will end with the keyword "**END**".

### Output

The output is the semantic version of the input HTML. In all converted tags you should **replace multiple spaces** (like <headerstyle="color:red">) with a single space and remove excessive spaces at the end (like **<footer >**). See the examples.

### Constraints

* Each line from the input holds either an HTML **opening tag** or an HTML **closing tag** or HTML **text content**.
* There will be no tags that span several lines and no lines that hold multiple tags.
* Attributes values will always be enclosed in **double quotes** **"**.
* Tags will never have **id** and **class** at the same time.
* The HTML will be **valid**. Opening and closing tags will match correctly.
* **Whitespace** may occur between attribute names, values and around comments (see the examples).
* Allowed working time: 0.1 seconds. Allowed memory: 16 MB.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| <div id="header">  </div> <!-- header -->  END | <header>  </header> |
| <div style="color:red" id="header">  </div> <!-- header -->  END | <header style="color:red">  </header> |
| <div class="header" style="color:blue">  </div> <!-- header -->  END | <header style="color:blue">  </header> |
| <div align="left" id="nav" style="color:blue">  <ul class="header">  <li id="main">  Hi, I have id="main".  </li>  </ul>  </div> <!-- nav -->  END | <nav align="left" style="color:blue">  <ul class="header">  <li id="main">  Hi, I have id="main".  </li>  </ul>  </nav> |
| <p class = "section" >  <div style = "border: 1px" id = "header" >  Header  <div id = "nav" >  Nav  </div> <!-- nav -->  </div> <!--header-->  </p> <!-- end paragraph section -->  END | <p class = "section" >  <header style = "border: 1px">  Header  <nav>  Nav  </nav>  </header>  </p> <!-- end paragraph section --> |