Тема: Regular Expressions
Обладнання: комп'ютери Pentium, Celeron.
Програмне забезпечення: Microsoft Visual Studio 2017, Microsoft Windows

Порядок виконання роботи:

Problem 1. Match Full Name

Write a regular expression to match a valid full name. A valid full name consists of **two words**, each word **starts** with a **capital letter** and contains **only lowercase letters afterwards**; each word should be **at least two letters long**; the two words should be **separated by a single space**.
To help you out, we've outlined several steps:

- Use an online regex tester like https://regex101.com/
- Check out how to use **character sets** (denoted with square brackets - "[]")
- Specify that you want two words with a space between them (the **space character ' '**, and not any whitespace symbol)
- For each word, specify that it should begin with an uppercase letter using a **character set**. The desired characters are in a range – **from 'A' to 'Z'**.
- For each word, specify that what follows the first letter are only **lowercase letters**, one or more – use another character set and the correct **quantifier**.
- To prevent capturing of letters across new lines, put "\b" at the beginning and at the end of your regex. This will ensure that what precedes and what follows the match is a word boundary (like a new line).

In order to check your regex, use these values for reference (paste all of them in the Test String field):

Examples

Match ALL of these	Match NONE of these
Ivan Ivanov	ivan ivanov, Ivan ivanov, ivan Ivanov, IVan Ivanov, Ivan IvAnov, Ivan Ivanov

Input	Output
ivan ivanov Ivan Ivanov end	Ivan Ivanov

Problem 2. Match Phone Number

Write a regular expression to match a valid phone number from Sofia. A valid number will start with "+359" followed by the area code (2) and then the number itself, consisting of 7 digits (separated in two group of 3 and 4 digits respectively). The different parts of the number are separated by **either a space or a hyphen** ('-'). Refer to the examples to get the idea.

- Use quantifiers to match a specific number of digits
- Use a capturing group to make sure the delimiter is **only one of the allowed characters (space or hyphen)** and not a combination of both. Use the group number to achieve this
- Add a word boundary at the end of the match to avoid partial matches (the last example on the right-hand side)
- Ensure that before the '+' sign there is either a space or the beginning of the string

Examples

Match ALL of these	Match NONE of these
+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

Input	Output
+359 2 222 2222 +3591345123 end	+359 2 222 2222
+359 2 234 5678 +359-2-234-5678 +359-2 234-5678 end	+359 2 234 5678 +359-2-234-5678

Problem 3. Series of Letters

Write a program that reads a string from the console and replaces all series of consecutive identical letters with a single one.

Examples

Input	Output
aaaaabbbbbcdddeeeedssaa	abcdedsa

Problem 4. Replace <a> tag

Write a program that replaces in a HTML document given as string **all the tags ...** with corresponding **tags [URL href=...>...[/URL]**. Read an input, until you receive **"end" command**. Print the result on the console.

Examples

Input	Output
SoftUni **end**	 [URL href=http://softuni.bg>SoftUni[/URL]

Note: The input may be read on a single line (unlike the example above) or from a file. Remove all new lines if you choose the first approach.

Problem 5. Extract Emails

Write a program to **extract all email addresses from a given text**. The text comes at the only input line. Print the emails on the console, each at 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. Examples of valid users: "**stephan**", "**mike03**", "**johnson**", "**st_steward**", "**softuni-bulgaria**", "**12345**". Examples of 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. Examples of hosts: "**bg**", "**software-university.com**", "**intoprogramming.info**", "**mail.softuni.org**". Examples of invalid hosts: "**helloworld**", "**.unknown.soft.**", "**invalid-host-**", "**invalid-**".

- Examples of **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.
- Examples of **invalid emails**: --123@gmail.com, …@mail.bg, .info@info.info, _steve@yahoo.cn, mike@helloworld, mike@.unknown.soft., johnson@invalid-.

Examples

Input	Output
Please contact us at: support@github.com.	*support@github.com*
Just send email to s.miller@mit.edu and j.hopking@york.ac.uk for more information.	*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.	*steve.parker@softuni.de*

Problem 6. 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 '**.**', '**!**' or '**?**'.

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.

Problem 7. * 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

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

	IelQ_Van4e

Problem 8. Semantic HTML

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

Examples

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