# Exercises: Regular Expressions (RegEx)

This document defines the **exercise assignments** for the ["Programming Fundamentals" course @ Software University](https://softuni.bg/courses/programming-fundamentals). Please submit your solutions (source code) of all below described problems in [Judge](https://judge.softuni.bg/Contests/430/Regex-Exercise).

## 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**", "**s.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: "**softuni.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.**, **s.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 |

## Extract Sentences by Keyword

Write a program that extracts **all sentences** that **contain** a particular **word** from a string(case-sensitive).

* Assume that the **sentences** are **separated** from each other by the character "." or "!" or "?".
* The **words** are separated by a **non-letter character**.
* Note that a **substring** is different than a **word**. The sentence “I am a fan of Mo**to**rhead” does not contain the word “**to**”. It contains the **substring** “**to**”, which is **not** what we need.
* Print the result text **without** the separators between the sentences ("." or "!" or "?").

### Examples

|  |
| --- |
| **Input** |
| **to**  Welcome **to** SoftUni! You will learn programming, algorithms, problem solving and software technologies. You need **to** allocate for study 20-30 hours weekly. Good luck! I am fan of Motorhead. To be or not **to** be - that is the question. TO DO OR NOT? |
| **Output** |
| Welcome **to** SoftUni  You need **to** allocate for study 20-30 hours weekly  To be or not **to** be - that is the question |

## Camera View

You are an amateur photographer and you want to calculate what will be seen in your pictures.

On the **first** line, you will receive an **array of integers** with exactly **two** elements:

**First** element – **m** will be the elements, which you have to skip. The **second element** – **n** will be the elements, which you have to **take**.

On the **next** line, you will receive a **string**, in which every camera will be marked with "**|<**". Skip the next **m** elements **immediately** after the camera and **take** the next **n** elements.

If you encounter **new** camera in the **view** 🡺 **stop** the current **camera** and **start** **new** **view** with the newly found.

### Output

Print **all** the taken **views** separated with ", ".

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 9 7  GreatBigSea|<uglyStuffHawaii|<boriiiingKilauea | Hawaii, Kilauea |
| 0 5  |>invalid|<beach|noMoreCameras | beach |

## Weather

You have to make a weather forecast about the weather depending on **strings**, which you receive from the **console**. Every string consists of **data** about the **city**, **average temperature** and **weather type**. You will receive strings **until** you receive the command “end”.

Every **valid** weather forecast **consists** of:

* **Two** **Latin** **capital** **letters**, which represent the code of the **city**
* **Immediately** followed by a **floating-point** number, which will represent the **average temperature**. Numbers **without** afloating point are **not** considered **valid**.
* Followed by the **type** of weather, which will consist of **uppercase** and **lowercase** **Latin** **letters**, starts **immediately** **after** the **temperature** and **ends** at the **first** occurrence of the sign ‘|’

If you receive input, which does **not** follow the rules above – **ignore** it.

If you receive a **new** **temperature** and/or type of weather for a city, which **already exists** – **rewrite** the previous values.

At the end, **print** the **temperature** and **weather type** for **every** city. **Order** the **cities** by **average** **temperature** in **ascending** **order**.

### Input

You will receive strings until you receive the command “end”.

### Output

Print **all** cities ordered by **average** **temperature** in **ascending** order. Use the following **format**:

“{nameOfTheCity} => {averageTemperature} => {typeOfWeather}”

**Format** the temperature to the **2nd decimal place**.

### Constraints

* The average temperature will be in the interval **[0.00…50.00]**
* The **floating-point** numbers will have at most **2** digits after the floating point.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| PB23.41Rainy|ASDASD  SDASCA20.21sUNNY|SDASD  asdaCA22.5rainy|sada  CA23.41cloydy  end | CA => 22.50 => rainy  PB => 23.41 => Rainy |

|  |  |
| --- | --- |
| **Input** | **Output** |
| invalidKA31.41|sunny|  validCA12.41Rainy|absad  gfASFasASPA31.21cloudy|asd  YA21.51sunny|  sadL21.41rainy|adas  end | CA => 12.41 => Rainy  YA => 21.51 => sunny  PA => 31.21 => cloudy |

## Key Replacer

You will be given a **key** **string** and a **text** **string**. The key string will contain a **start** **key** and an **end** **key**.

The **start** **key** starts at the **beginning** of the **string** and **ends** at the **first** occurrence of one of the symbols – “|”, “<” or “\”. The **end** **key** starts at the **last** occurrence of **one** of **these** **symbols** and **ends** when the **string** **ends**. Both keys can contain **only** **Latin** **alphabet** **letters**.

When you extract **both** keys search for them in the text string and extract every string, which is **between** them. **Concatenate** all **collected** **strings** and **print** the **result**. If the result string is **empty** print “Empty result”.

### 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

Print the **concatenated** **message**, if such exists or "**Empty result**", if it does not.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| start<213asfaas|end  saaastarthelloendsdarstartFromTheOtherenddvsefdsfstartSideend | helloFromTheOtherSide |

|  |  |
| --- | --- |
| **Input** | **Output** |
| A|safafasfsadf|B  NoMEssageABhereYeyAB | Empty result |

## \* Valid Usernames

You are part of the back-end development team of the next Facebook.

You are given a **line of usernames**, **separated** by one of the following symbols: “ ”, “/”, “\”, “(“, “)”**.**

First you have to export all **valid** usernames. A valid username **starts with a letter** and can onlycontain **letters**, **digits** andunderscores “\_”. 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 the 2 valid usernames with **biggest** **sum** of their **lengths,** on the console, 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:“ “, “/”, “\”, “(“, “)”.

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

### Output

Print the 2 **consecutive** **valid usernames** with the **biggest** **sum** of their lengths on the console, each on a separate line.

If there are **2 or more couples** of usernames with the same sum of their lengths, print the **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 **field=value** pairs. 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** |
| login=student&password=student  END |
| **Output** |
| 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] |