# Exercises: Strings and 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/321/Strings-and-RegEx-Exercise).

# Strings

## Convert from base-10 to base-N

Write a program that takes a base-10 number (0 to 1050) and converts it to a base-N number, where 2 <= N <= 10.  
The input consists of 1 line containing two numbers separated by a single space. The first number is the base N to which you have to convert. The second one is the base 10 number to be converted. **Do not use any built in converting functionality, try to write your own algorithm.**

### Examples

|  |  |
| --- | --- |
| **Base-10** | **Base-N** |
| 7 10 | 13 |
| 3 154 | 12201 |
| 5 123 | 443 |
| 4 1000 | 33220 |
| 9 3487 | 4704 |

## Convert from base-N to base-10

Write a program that takes a base-N number and converts it to a base-10 number (0 to 1050), where 2 <= N <= 10.  
The input consists of 1 line containing two numbers separated by a single space. The first number is the base N to which you have to convert. The second one is the base N number to be converted. **Do not use any built in converting functionality, try to write your own algorithm.**

### Examples

|  |  |
| --- | --- |
| **Base-N** | **Base-10** |
| 7 13 | 10 |
| 3 12201 | 154 |
| 5 443 | 123 |
| 4 33220 | 1000 |
| 9 4704 | 3487 |

## Sum big numbers

You are given two lines - each can be a really big number (0 to 1050). You must display the sum of these numbers.

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

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Input** | **Output** |
| 23  23 | 46 | 9999  1 | 10000 |
| **Input** | | | | **Output** |
| 923847238931983192462832102  934572893617836459843471846187346 | | | | 934573817465075391826664309019448 |

## Multiply big number

You are given two lines - the first one can be a really big number (0 to 1050). The second one will be a single digit number (0 to 9). You must display the product of these numbers.

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

### Examples

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | **Output** | **Input** | **Output** | **Input** | **Output** |
| 23  2 | 46 | 9999  9 | 89991 | 923847238931983192462832102  4 | 934573817465075391826664309019448 |

## Unicode Characters

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

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Hi! | \u0048\u0069\u0021 |
| What?!? | \0057\0068\0061\0074\003f\0021\003f |
| SoftUni | \0053\006f\0066\0074\0055\006e\0069 |

## Character Multiplier

Create a **method** that takes two strings as arguments and returns the sum of their character codes multiplied (multiply str1.charAt (0) with str2.charAt (0) and add to the total sum). Then continue with the next two characters. If one of the strings is longer than the other, add the remaining character codes to the total sum without multiplication.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Gosho Pesho | 53253 |
| 123 522 | 7647 |
| a aaaa | 9700 |

## Magic exchangeable words

Write a **method** that takes as input two strings, and returns Boolean if they are exchangeable or not. 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 "**aabbccbb"** and "**nnooppzz"** 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 '**l'** 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 |

## \*Letters Change Numbers

**This problem is from the Java Basics exam (8 February 2015). You may check your solution** [here](https://judge.softuni.bg/Contests/Practice/Index/69#1)**.**

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 |  |

## \*\* 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.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| astalavista baby  sta | Shaked it.  No shake.  alavi baby |

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

# RegEx

## 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](mailto:info-bg@software-university.software.academy).
* Examples of **invalid emails**: [--123@gmail.com](mailto:--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-).

### 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 from a text all sentences that contain a particular word** (case-sensitive).

* Assume that the **sentences** are separated from each other by the character "." or "!" or "?".
* The **words** are separated one from another by a **non-letter character**.
* Notе that appearance as **substring** is different than appearance as **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 **sentence text** without the separators between the sentences ("." or "!" or "?").

### Example

|  |
| --- |
| **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 |

## \*Valid Usernames

**This problem is from the Java Basics Exam (21 September 2014 Evening). You may check your solution** [**here**](https://judge.softuni.bg/Contests/Practice/Index/34#2)**.**

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

**This problem is originally from the JavaScript Basics Exam (22 November 2014). You may check your solution** [**here**](https://judge.softuni.bg/Contests/Practice/Index/84#3)**.**

**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] | |

## \*Use Your Chains, Buddy

**This problem is from the JavaScript Basics Exam (9 January 2015). You may check your solution** [**here**](https://judge.softuni.bg/Contests/Practice/Index/56#2)**.**

You are in Cherny Vit now and there are 12km to Anchova Bichkiya Hut. You need to get there by car. But there is so much snow that you need to use car chains. In order to put them on the wheels correctly, you need to read the manual. But it is encrypted…

As input you will receive **an HTML document** as a **single string**. You need to get the text from **all the <p> tags** and replace all characters which are **not small letters and numbers** with a space **" "**. After that you must decrypt the text – all letters **from a to m** are **converted** to letters **from n to z** (a  n; b  o; … m  z). The letters **from n to z** are **converted** to letters **from a to m** (n  a; o  b; … z  m). All **multiple** spaces should then be replaced by only **one space**.

For example, if we have **"<div>Santa</div><p>znahny # grkg ()&^^^&12</p>"** we extract **"znahny # grkg ()&^^^&12"**. Every **character** that is **not a small letter or a number** is replaced with aspace (**"znahny grkg 12"**). We convert each small letter as described above (**"znahny grkg 12"  "manual text 12"**) and replace all multiple spaces with only **one space** (**"manual text 12"**). Finally, we concatenate the decrypted text from all **<p></p>** tags (in this case, it's only one). And there you go – you have the manual ready to read!

### Input

The input is read from the console and consists of just one line – the **string** with the **HTML document**.

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

### Output

**Print** on a single line the decrypted text of the manual. See the given **examples** below.

### Constraints

* Allowed working time: 0.2 seconds. Allowed memory: 16 MB.

### Examples

|  |
| --- |
| **Input** |
| <html><head><title></title></head><body><h1>hello</h1><p>**znahny!@#%&&&&\*\*\*\***</p><div><button>dsad</button></div><p>**grkg^^^^%%%)))([]12**</p></body></html> |
| **Output** |
| manual text 12 |
|  |
| **Input** |
| <html><head><title></title></head><body><h1>Intro</h1><ul><li>Item01</li><li>Item02</li><li>Item03</li></ul><p>**jura qevivat va jrg fyvccrel fabjl**</p><div><button>Click me, baby!</button></div><p> **pbaqvgvbaf fabj qpunvaf ner nofbyhgryl rffragvny sbe fnsr unaqyvat nygubhtu fabj punvaf znl ybbx** </p><span>This manual is false, do not trust it! The illuminati wrote it down to trick you!</span><p>**vagvzvqngvat gur onfvp vqrn vf ernyyl fvzcyr svg gurz bire lbhe gverf qevir sbejneq fybjyl naq gvtugra gurz hc va pbyq jrg**</p><p> **pbaqvgvbaf guvf vf rnfvre fnvq guna qbar ohg vs lbh chg ba lbhe gverf**</p></body> |
| **Output** |
| when driving in wet slippery snowy conditions snow dchains are absolutely essential for safe handling although snow chains may look intimidating the basic idea is really simple fit them over your tires drive forward slowly and tighten them up in cold wet conditions this is easier said than done but if you put on your tires |