Exercises: Manual 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.

Problem 1. Reverse String

Write a program that reads a string from the console, reverses it and prints the result back at the console.

Examples

Input	Output
sample	elpmas
24tvcoi92	29iocvt42

Problem 2. String Length

Write a program that reads from the console a string of maximum 20 characters. If the length of the string is less than 20, the rest of the characters should be filled with *. Print the resulting string on the console.

Examples

Input	Output
Welcome to SoftUni!	Welcome to SoftUni!*
a regular expression (abbreviated regex or regexp and sometimes called a rational expression) is a sequence of characters that forms a search pattern	a regular expression
C#	C#******************

Problem 3. Formatting Numbers

Write a program that reads 3 numbers (separated by whitespace): an integer **a** (0 ≤ **a** ≤ 2222), a floating-point **b** and a floating-point **c** and **prints them in 4 virtual columns** on the console. Each column should have a width of 10 characters. The number **a** should be printed in **hexadecimal, left aligned**; then the number **a** should be printed in binary form, padded with zeroes (if it is bigger than 10 bits remove the least significant ones), then the number **b** should be **printed with 2 digits after the decimal point, right aligned**; the number **c** should be **printed with 3 digits after the decimal point, left aligned**.

Examples

a	b	c	result
254	11.6	0.5	|FE |0011111110| 11.60|0.500 |
499	-0.5559	10000	|1F3 |0111110011| -0.56|10000.000 |
0	3	-0.1234	|0 |0000000000| 3.00|-0.123 |
444	-7.5	7.5	|1BC |0110111100| -7.50|7.500 |

Problem 4. Convert from base-10 to base-N

Write a program that takes a base-10 number (0 to 10^{50}) 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

Problem 5. 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 10^{50}), 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

Problem 6. 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 is an integer number. Please ignore the **character casing**. **Overlapping** between occurrences is **allowed**.

Examples

Input	Output
Welcome to the Software University (SoftUni)! Welcome to programming. Programming is wellness for developers, said Maxwell. wel	4
aaaaaa	5

aa	
ababa caba aba	3
Welcome to SoftUni Java	0

Problem 7. 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.

Examples

Input	Output	Input	Output
23 23	46	9999 1	10000

Input	Output
923847238931983192462832102 934572893617836459843471846187346	93457381746507539182666430901944 8

Problem 8. Multiply big number

You are given two lines - the first one can be a really big number (0 to 10^{50}). 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	93457381746507539182666430901944 8

Problem 9. Text Filter

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

them by calling the OS GNU/**Linux**! Sincerely, a **Windows** client	Sincerely, a ******* client

Problem 10. 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

Problem 11. Palindromes

Write a program that extracts from a given text all palindromes, e.g. ABBA, lamal, exe and prints them on the console on a single line, separated by comma and space. Use spaces, commas, dots, question marks and exclamation marks as word delimiters. Print only **unique** palindromes, **sorted** lexicographically (small letters are before big ones).

Examples

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

TIP: if you have problems with the sorting - take a look at the Collator class

Problem 12. 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

Problem 13. 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

Problem 14. * Letters Change Numbers

This problem is from the Java Basics exam (8 February 2015). You may check your solution here.

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	

Problem 15. ** 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. ##!!.*.#

Problem 16. *** 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
 END	http://softuni.bg
This text has no links END	
<!DOCTYPE html> <html> <head> <title>Hyperlinks</title> <link href="theme.css" rel="stylesheet" /> </head> <body> Home Courses Forum Forum <a id="js" href =	/ /courses /forum # javascript:alert('hi yo') http://www.nakov.com #empty # #commented

```
"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
```