Strings, Dictionaries, Lambda and LINQ

1. Reverse String

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

Examples

Input	Output
sample	elpmas
24tvcoi92	29iocvt42

Hints

- **Variant I**: convert the string to **char array**, **reverse** it, then convert it to **string** again.
- **Variant II**: print the letters of the string in back direction (from the last to the first) in a **for**-loop.

2. Fit String in 20 Chars

Write a program that **reads** from the console a string and **fits the string in 20 characters** as follows:

- If the string has **less than 20 characters**, append some '*****' until it gets length of exactly 20 characters.
- If the string length is **more than 20 characters**, discard all characters after the first 20.

Print the result string at the console.

Examples

Input	Output
Welcome to SoftUni!	Welcome to SoftUni!*
A "regular expression" (abbreviated regex or regexp) is a sequence of characters that forms a search pattern.	a regular expression
C#	C#******************

Hints

- If string **`length`** < **`20`**, use **`PadRight(20, '*')`**.
- If string **`length`** > **`20`**, use **`Substring(0, 20)`**.

3. Censor Your Email Address

You have some text that contains your email address. You're sick of spammers, so you want to **hide** it. You decide to **censor** your email: to **replace all characters** in it with asterisks ('*') **except the domain**.

Assume your email address will always be in format **[username]@[domain]**. You need to replace the username with asterisks of equal number of letters and keep the domain unchanged.

Input

- The first line holds your **email** address.
- The second line holds a **text** where the email should be censored.

Examples

Input
pesho.peshev@email.bg My name is Pesho Peshev. I am from Sofia, my email is: pesho.peshev@email.bg (not pesho.peshev@email.com). Test: pesho.meshev@email.bg, pesho.peshev@email.bg
Output
My name is Pesho Peshev. I am from Sofia, my email is: **********@email.bg (not pesho.peshev@email.com). Test: pesho.meshev@email.bg, **********@email.bg

Hints

In order to accomplish the task, you may find these steps useful:

- **Split** the email into two parts – **username** and **domain**.
- Create the **replacement** string by duplicating the '*****' character **`username.Length`** and appending '**@**' and the **domain**.
- **Replace** all occurrences of your **email** with the **replacement string**.

4. 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**.
- Note 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

Hints

- First **extract the sentences** (just split by '**.**', '**!**' and '**?**').
- **Split each sentence into words**. How? Replace each non-letter character with space. Then split by space and remove all empty tokens.
- Finally, **check** whether the **target word** occurs in the list of words found in each sentence.

5. URL Parser

Write a program that **parses an URL** given in the following format:

```
[protocol]://[server]/[resource]
```

The parsing extracts its parts: protocol, server and resource.

- The **[server]** part is mandatory.
- The **[protocol]** and **[resource]** parts are optional.

Examples

Input	Output

`http://www.abc.com/video`	`[protocol] = "http"`
	`[server] = "www.abc.com"`
	`[resource] = "video"`

`https://www.softuni.bg/Resources/Materials`	`[protocol] = "https"`
	`[server] = "www.softuni.bg"`
	`[resource] = "Resources/Materials"`

`ftp://www.su.us/TestResource`	`[protocol] = "ftp"`
	`[server] = "www.su.us"`
	`[resource] = "TestResource"`

`https://softuni.bg`	`[protocol] = "https"`
	`[server] = "softuni.bg"`
	`[resource] = ""`

`www.nakov.com`	`[protocol] = ""`
	`[server] = "www.nakov.com"`
	`[resource] = ""`

Hints

- Find the leftmost occurrence of "**://**" in the input URL.
  - If **found**, the left side holds the **protocol**, the right side holds the **server + resource**.
  - If **not found**, the protocol is missing, the input string holds **server + resource** only.
- After the "protocol" part is removed from the input URL, find the leftmost occurrence of "**/**".
  - If **found**, the left side holds the **server**, the right side holds the **resource**.
  - If **not found**, the resource is missing, the whole string holds the **server**.

6. * Reverse the Words in a Sentence

Write a program that **reverses the words in a given sentence** without changing the **punctuation and spaces**.

- Use the following **separators** between the words: **. , : ; = ( ) & [ ] " ' \ / ! ?** *(space)*.
- All **other characters** are considered part of words, e.g. **C++**, **a+b**, and **a77** are considered valid words.
- The **sentences** always **start by word** and **end by separator**.

Examples

Input	Output
`C# is not C++, and PHP is not Delphi!`	`Delphi not is PHP, and C++ not is C#!`
`The quick brown fox jumps over the lazy dog /Yes! Really!!!/.`	`Really Yes dog lazy the over jumps fox brown /quick! The!!!/.`
`Pack my box (with 5 dozen liquor jugs).`	`jugs liquor dozen (5 with box my Pack).`
`var separators = sentence.Split(letters, StringSplitOptions.RemoveEmptyEntries);`	`RemoveEmptyEntries StringSplitOptions = letters.Split(sentence, separators.var);`

Hints

- **Extract all words** by splitting by the specified **separator chars** and removing the empty tokens.
- Append all words to **obtain all word characters** (characters in the sentence that are non-separators).
- **Split by all word characters** to obtain all **separator strings** between the words.
- **Reverse the words**.
    - Now we have two lists: **reversed words** and **separator strings** (coming after each original word).
    - We need to join the reversed words with the separators.
- Print the results as follows: **first word**, **first separator string**, **second word**, **second separator string**, …

sentence	var separators = sentence.Split(letters, StringSplitOptions.RemoveEmptyEntries);						
words	var	separators	sentence	Split	letters	StringSplitOptions	RemoveEmptyEntries
separator strings	" "	" = "	"."	"("	", "	"."	";"
words reversed	RemoveEmpty Entries	StringSplit Options	letters	Split		sentence	separators
sentence reversed	RemoveEmptyEntries StringSplitOptions = letters.Split(sentence, separators.var);						

7. Change to Uppercase

We are given a text. Write a program that **modifies the casing of letters to uppercase** at all places **in the text surrounded by <upcase> and </upcase> tags**. Tags cannot be nested.

Example

Input
Welcome to the **<upcase>Software University</upcase>.** Learn **<upcase>computer programming</upcase>** and start a **<upcase>job</upcase>** in a software company.
Output
Welcome to the **SOFTWARE UNIVERSITY.** Learn **COMPUTER PROGRAMMING** and start a **JOB** in a software company.

Hints

- You may find the position of the first **<upcase>** and the first **</upcase>**, delete the text between and insert the uppercase version of the text without the tags at the position of **<upcase>**.
- Repeat the above until no more **<upcase>** and **</upcase>** tags are found in the text.

8. 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**.
- All words are processes are **case-sensitive**.
- Print all **unique** palindromes (no duplicates), **sorted** lexicographically.

Examples

Input	Output

Hi,exe? ABBA! Hog fully a string: ExE. Bob	a, ABBA, exe, ExE

9. Capitalization

Write a program which takes input string and **capitalizes the first character of each word** and does not affect the others. Use the standard separators between words: *(space)*, "**.**", "**,**", "**?**", "**!**", "**;**".

Examples

Input	Output
jon skeet	Jon Skeet
old mcdonald	Old Mcdonald
miles o'Brien	Miles O'Brien

Hints

- **Split input string by the separators** and store all words in array.
- Get every word's **first character** and transform it to uppercase using char.ToUpper(symbol).
- **Make a new substring** from index 1 to end of the word.
- **Concatenate** the new first character with the substring and print the newly formed word.
- Another approach is to use TextInfo.ToTitleCase(string).

10. Palindrome Index

Given a string of lowercase letters, determine the **index** of the character **whose removal will make the string a palindrome**. If the string is **already a palindrome**, then print **-1**. There will always be a valid solution.

Examples

Input	Output	Comments
aaab	3	If we remove letter "**b**" at index **3** we will get a palindrome "**aaa**".
baa	0	Remove "**b**" at index **0** to get a palindrome "**aa**".
aaa	-1	"**aaa**" is already a palidrome.

11. Common Strings

You are given two strings, **A** and **B**. Find if there is a substring that appears in **both A and B**.

Examples

Input	Output	Comments
hello world	yes	The letter "o" is common between both strings, hence the output is "yes". Furthermore, the letter "l" is common, but you only need 1 common substring.
hi world	no	Both words do not have common substring.
soft softuni	yes	Substring "soft" is common between both strings.

12. Phonebook

Write a program that receives some info from the console about **people** and their **phone numbers**. Each **entry** should have just **one name** and **one number** (both of them strings).

On each line you will receive some of the following commands:

- **A {name} {phone}** – adds entry to the phonebook. In case of trying to add a name that is already in the phonebook you should change the existing phone number with the new one provided.
- **S {name}** – searches for a contact by given name and prints it in format "**`{name} -> {number}`**". In case the contact isn't found, print "**`Contact {name} does not exist.`**".
- **END** – stop receiving more commands.

Examples

Input	Output
A Nakov 0888080808 S Mariika S Nakov END	Contact Mariika does not exist. Nakov -> 0888080808
A Nakov +359888001122 A RoYaL(Ivan) 666 A Gero 5559393 A Simo 02/987665544 S Simo S simo S RoYaL S RoYaL(Ivan) END	Simo -> 02/987665544 Contact simo does not exist. Contact RoYaL does not exist. RoYaL(Ivan) -> 666
A Misho +359883123 A Misho 02/3123 S Misho END	Misho -> 02/3123

Hints

- **Parse the commands** by splitting by space. Execute the commands until "**END**" is reached.
- Store the **phonebook entries** in **Dictionary<string, string>** with key **{name}** and value **{phone number}**.

13. Phonebook Upgrade

Add functionality to the phonebook from the previous task to **print all contacts ordered lexicographically** when receive the command "**`ListAll`**".

Examples

Input	Output
A Nakov +359888001122 A RoYaL(Ivan) 666 A Gero 5559393 A Simo 02/987665544 ListAll END	Gero -> 5559393 Nakov -> +359888001122 RoYaL(Ivan) -> 666 Simo -> 02/987665544

Hints

- **Variant I (slower):** Sort all entries in the dictionary by key and print them.
- **Variant II (faster):** Keep the entries in more appropriate data structure that will keep them in sorted order for better performance.