# Lab: Text Processing and Regular Expressions

Problems for exercises and homework for the ["Technology Fundamentals with C#" course @ SoftUni](https://softuni.bg/modules/57/tech-module-4-0).

You can check your solutions in [Judge](https://judge.softuni.bg/Contests/1216).

## Reverse Strings

You will be given series of strings until you receive an **"end"** command. Write a program that reverses strings and prints each pair on a separate line in the format **"{word} = {reversed word}"**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| helLo  Softuni  bottle  end | helLo = oLleh  Softuni = inutfoS  bottle = elttob |
| Dog  caT  chAir  end | Dog = goD  caT = Tac  chAir = riAhc |

### Solution

Use a while loop and read strings until you receive "end".



Reverse the string with a for loop. Start from the last index and append each symbol to the new string.



Print the reversed string in the specified format.



## Repeat Strings

### Write a program that reads an array of strings. Each string is repeated N times, where N is the length of the string. Print the concatenated string.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| hi abc add | hihiabcabcabcaddaddadd |
| work | workworkworkwork |
| ball | ballballballball |

### Solution

* Read a string array.



* Initialize **StringBuilder**.



* Iterate through elements in the array.



* Find the length of the current word and append it.



* Print the **StringBuilder**.

## Substring

On the **first line** you will receive a **string**. On the **second line** you will receive a second **string**. Write a program that **removes** **all** of the **occurrences** of the **first** string **in** the **second,** **until** there is **no match**. At the end **print** the **remaining string**.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comment** |
| Ice  kicegiciceeb | kgb | We remove ice once and we get "kgiciceeb"  We match "ice" one more time and we get "kgiceb"  There is one more match. The finam result is "kgb" |

### Hints

* Read the input.
* Find the first index where the key appears.
  + Use the built-in method **IndexOf**()
* Remove the match.
  + Use the built-in method **Remove**(index, length)
* Repeat it until the text doesn't contain the key anymore.

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

### Hints

* Read the input.
* Replace all ban words in the text with asterisk (\*).
  + Use the built-in method **Replace**(banWord, replacement).
  + Use **new string(**char ch, int repeatCount**)** to create the replacement

## Digits, Letters and Other

Write a program that receives a **single** **string** and on the **first** **line,** prints **all the digits**. On the **second** – **all the letters**, and on the **third** – **all the other characters**. **There will always be at least one digit, one letter and one other character.**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Agd#53Dfg^&4F53 | 53453  AgdDfgF  #^& |

### Hints

* Read the input.
* Use loop to iterate through all characters in the text. If the char is digit print it, otherwise ignore it.
  + Use char.**IsDigit**(char symbol)
* Do the same for the letters and other chars
  + Find something like IsDigit method for the letters.

Problems for exercises and homework for the ["Technology Fundamentals" course @ SoftUni](https://softuni.bg/courses/technology-fundamentals).  
Please submit your solutions (source code) of all below described problems in the [Judge System.](https://judge.softuni.bg/Contests/1344/Regular-Expressions-Lab)

## Match Full Name

Write a program that **matches full names** from a list of names and **prints** them on the console.

### Writing the Regular Expression

First, write a regular expression to match a valid full name, according to these conditions:

* A valid full name has the following characteristics:
  + It consists of **two words**.
  + Each word **starts** with a **capital letter**.
  + After the first letter, it **only contains lowercase letters afterwards**.
  + **Each** of the **two words** should be **at least two letters long**.
  + The **two words** are **separated** by a **single space**.

To help you out, we've outlined several steps:

* Use an online regex tester like <https://regex101.com/>
* Check 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 the 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):

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

By the end, the matches should look something like this:



After you’ve constructed your regular expression, it’s time to write the solution in C#.

### Examples

|  |
| --- |
| **Input** |
| Ivan Ivanov, Ivan ivanov, ivan Ivanov, IVan Ivanov, Test Testov, Ivan Ivanov |
| **Output** |
| Ivan Ivanov Test Testov |