# Lab: Strings and Text Processing

You can check your solutions in [Judge](https://alpha.judge.softuni.org/contests/text-processing-lab/1216/practice)

## Reverse Strings

You will be given a 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 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

Create 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 the elements of 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**. Create 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" |
| hep  ShepoftunihepIsGrhepeat | SoftuniIsGreat |  |

### 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

Create a program that takes a **text** and a **string of banned words**. All words included in the ban list should be replaced with a string of **asterisks** '**\***', whose length must be 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 |
| von Richthofen, German, 80 air  Manfred Albrecht Freiherr **von Richthofen**, known in English as Baron **von Richthofen** was a fighter pilot with the **German** Air Force during World War I. He is considered the ace-of-aces of the war, being officially credited with **80 air** combat victories. | Manfred Albrecht Freiherr \*\*\*\*\*\*\*\*\*\*\*\*\*\*, known in English as Baron \*\*\*\*\*\*\*\*\*\*\*\*\*\* was a fighter pilot with the \*\*\*\*\*\* Air Force during World War I. He is considered the ace-of-aces of the war, being officially credited with \*\*\*\*\*\* combat victories. |

### Hints

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

## Digits, Letters and Others

Create a program that receives a **single** **string** and prints **all the digits** on the **first** **line**, 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  #^& |
| So%f94t34U\*n&i></37 | 943437  SoftUni  %\*&></ |

### Hints

* Read the input.
* Use a 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 the **IsDigit** method for the letters.