# Exercise: Strings and Text Processing

Problems for exercise and homework for the ["C# Fundamentals" course @ SoftUni](https://softuni.bg/trainings/3606/programming-fundamentals-with-csharp-january-2022)  
You can check your solutions here: [Judge](https://judge.softuni.org/Contests/1217/Text-Processing-Exercise)

## Valid Usernames

Create a program that **reads user** names on a **single** line (joined by "**,** ") and **prints** all **valid usernames**.

A valid username:

* Has **length** between 3 and 16 characters and
* **Contains** only **letters**, **numbers**, **hyphens**, and **underscores**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| sh, too\_long\_username, !lleg@l ch@rs, jeffbutt | jeffbutt |
| Jeff, john45, ab, cd, peter-ivanov, @smith | Jeff  John45  peter-ivanov |

## Character Multiplier

Create a **method** that takes **two strings** as arguments and returns the **sum** of their **character codes** **multiplied** (multiply **str1[0]** with **str2[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** |
| Peter George | 52114 |
| 123 522 | 7647 |
| a aaaa | 9700 |

## Extract File

Create a program that reads the path to a file and **subtracts** the **file name** and its **extension**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| C:\Internal\training-internal\Template.pptx | File name: Template  File extension: pptx |
| C:\Projects\Data-Structures\LinkedList.cs | File name: LinkedList  File extension: cs |

## Caesar Cipher

Create a program that returns an **encrypted version** of the same text. Encrypt the text by **shifting** **each character** with **three** positions **forward**. For example, **A** would be replaced by **D**, **B** would become **E**, and so on. Print the **encrypted** **text**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Programming is cool! | Surjudpplqj#lv#frro$ |
| One year has 365 days. | Rqh#|hdu#kdv#698#gd|v1 |

## 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)**. Your task is to display the product of these numbers.

Note: do not use the BigInteger class.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 23  2 | 46 |
| 9999  9 | 89991 |
| 923847238931983192462832102  4 | 3695388955727932769851328408 |

## Replace Repeating Chars

Create a program that reads a string from the console and **replaces** any **sequence of the same letters** with a **single** **corresponding letter**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| aaaaabbbbbcdddeeeedssaa | abcdedsa |
| qqqwerqwecccwd | qwerqwecwd |

## String Explosion

Explosions are marked with **'**>**'**. Immediately after the mark, there will be an **integer**, which signifies the **strength** of the explosion.

You should **remove** x **characters** (where x is the **strength** of the explosion), **starting after** the punched **character** (**'**>**'**).

If you find **another** explosion mark (**'**>**'**) while you’re deleting characters, you should **add** the **strength** to your **previous** **explosion**.

When all characters are processed, **print** the string **without** the **deleted** **characters**.

You should **not** delete the **explosion** character – **'**>**'**, but you should **delete** the **integers**, which represent the **strength**.

### Input

You will receive a **single** **line** with the string.

### Output

Print what is left from the string after explosions.

### Constraints

* You will **always** receive **strength** for the punches
* The path will consist only of letters from the **Latin** **alphabet**, **integers,** and the char **'**>**'**
* The strength of the punches will be in the interval [0…9]

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| abv>1>1>2>2asdasd | abv>>>>dasd | 1st explosion is at index **3** and it is with a **strength** of **1**. We delete **only** the **digit** **after** the explosion character. The string will look like this: abv>>1>2>2asdasd  2nd explosion is with strength **one** and the string transforms to this: abv>>>2>2asdasd  3rd explosion is now with a strength of 2. We delete the digit and we find **another** explosion. At this point, the string looks like this: abv>>>>2asdasd.  4th explosion is with strength **2**. We have **1** strength **left** from the previous explosion, we **add** the strength of the **current** explosion to what is **left,** and that adds up to a **total** strength of **3**. We **delete** the next **three** **characters** and we **receive** the **string** abv>>>>dasd  We do **not** have **any more explosions** and we print the result: abv>>>>dasd |
| peter>2sis>1a>2akarate>4hexmaster | peter>is>a>karate>master |  |