**Lab: Text Processing**

Problems for exercises and homework for the ["Programming Fundamentals" course @ HYPERLINK "https://softuni.bg/trainings/3951/programming-fundamentals-with-java-january-2023"SoftUni](https://softuni.bg/trainings/3951/programming-fundamentals-with-java-january-2023)

You can check your solutions in [Judg HYPERLINK "https://judge.softuni.org/Contests/1669/Text-Processing-Lab"e HYPERLINK "https://judge.softuni.org/Contests/1669/Text-Processing-Lab".](https://judge.softuni.org/Contests/1669/Text-Processing-Lab)

* **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 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 word at each iteration and append it to the **StringBuilder**.



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 final result is "kgb" |
| e  fixture | fixtur |  |

**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 **replace(String oldValue, String newValue)**
* 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 |
| computer, programming, set  In **computer** **programming**, an application **programming** interface (API) is a **set** of subroutine definitions, communication protocols, and tools for building software. | In \*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*, an application \*\*\*\*\*\*\*\*\*\*\* interface (API) is a \*\*\* of subroutine definitions, communication protocols, and tools for building software. |

**Hints**

* Read the input.
* Replace all ban words in the text with an asterisk (\*).
* Use the built-in method **replace(banWord, 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 another character.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| Agd#53Dfg^&4F53 | 53453  AgdDfgF  #^& |
| a1! | 1  a  ! |

**Hints**

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