# Exercises: Strings and Text Processing

Problems for exercises and homework for the [“Programming Fundamentals Extended” course @ SoftUni](https://softuni.bg/courses/programming-fundamentals).

Check your solutions here: <https://judge.softuni.bg/Contests/441>.

## Placeholders

Placeholders in C# are really comfortable for use, right? How about you implement them to see just how easy they are.

You will be given input lines containing a string with placeholders, i.e. “This is {0}. And that is {1}.”.

The input lines will be in the following format:

{someString} -> {someElement}, {someElement2}

You have to **replace** the **placeholders** in the **string**, with the **elements** **given after that**, **separated** by a **comma** and a **space**. **Each** **element** has an **index**, so you must give them the **right order**. The **first given element** goes where the **0** is, the **second** to the **1**, and so on…

Print **each** **string**, with its **replaced** **placeholders**, **RIGHT after you’ve read it**, and **BEFORE reading** the **next one**.

The input ends when you receive the command “**end**”.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| I have a {0} and a {1} -> car, house  Cool, and I have a {0}. -> yacht  Darn... You beat me {0} {1} -> with, this  end | I have a car and a house  Cool, and I have a yacht.  Darn... You beat me with this |
| This is {2} {1} {0}. -> ment, I, what  And this is what you ment. -> nothing  end | This is what I ment.  And this is what you ment. |

### Hint

* Due to the fact that the sentences hold spaces, you cannot split the sentence and the values, by space. There should be functionality in C# which removes the border spaces from a string, if you split it by   
  “->”, though.
* You might have to **split first** the **sentence** and the **values**, and **then the values**, because their **delimiter** **contains** a **space**.

## JSON stringify

JSON is a syntax for storing and exchanging data. It comes from **J**ava**S**cript **O**bject **N**otation. You have been tasked to create a database of students, and stored them in a JSON format.

First, create a **class Student**, which will represent our **data** **model** – it will hold our **data**. The **Student** should have **Name** (**string**), **Age** (**integer**), and **Grades** (**Collection** of **integer numbers**).

This will be enough to store our data. We’ll need a collection of Students, since we will store a lot of students.

And now let’s see the input and output formats.

The input will consist of several input lines in the following format:

{name} : {age} -> {grade1}, {grade2}, {grade3}. . .

Store every student with its **name**, **age** and **given grades**. The **name** will be a **string** which can contain **any** **ASCII** characters except **space** (“ “), “:”, “-”, “>”, “,”. The **age** will be an **integer**. The **grades** will be **integer numbers**.

When you receive the command “**stringify**”, the input sequence ends.

You should print the **collection of students** in the following format: [{student1},{student2}. . .]

Each student must be printed in the following format:

{name:”{name}”,age:{age},grades:[{grade1}, {grade2}. . .]}

See the examples for more info.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Pesho : 25 -> 6, 6, 5  Gosho : 10 -> 3, 3, 4  Ivan : 28 -> 2, 2, 3  stringify | [{name:"Pesho",age:25,grades:[6, 6, 5]},{name:"Gosho",age:10,grades:[3, 3, 4]},{name:"Ivan",age:28,grades:[2, 2, 3]}] |
| Johnny : 11 -> 6, 4, 3  Peter : 10 -> 5, 5, 4  Jordan : 13 -> 6, 6  Donald : 10 ->  Isacc : 20 -> 2, 2, 3  Alex : 11 -> 6, 6, 3  stringify | [{name:"Johnny",age:11,grades:[6, 4, 3]},{name:"Peter",age:10,grades:[5, 5, 4]},{name:"Jordan",age:13,grades:[6, 6]},{name:"Donald",age:10,grades:[]},{name:"Isacc",age:20,grades:[2, 2, 3]},{name:"Alex",age:11,grades:[6, 6, 3]}] |

## JSON parse

Stringifying a JSON was easy? You want something harder?

Well … JSON parse is your problem. You will receive a JSON collection of students. You need to parse it to Student objects.

The input will consist of a single line containing the students’ info. The students will be given in the following way:

[{student1},{student2}. . .]

Each student will be given in the following format:

{name:”{name}”,age:{age},grades:[{grade1}, {grade2}. . .]}

Parse the input data, and print each student in the following way.

{name} : {age} -> {grade1}, {grade2}, {grade3}. . .

In case there are **NO grades**, print “None”.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| [{name:"Pesho",age:25,grades:[6, 6, 5]},{name:"Gosho",age:10,grades:[3, 3, 4]},{name:"Ivan",age:28,grades:[2, 2, 3]}] | Pesho : 25 -> 6, 6, 5  Gosho : 10 -> 3, 3, 4  Ivan : 28 -> 2, 2, 3  stringify |
| [{name:"Johnny",age:11,grades:[6, 4, 3]},{name:"Peter",age:10,grades:[5, 5, 4]},{name:"Jordan",age:13,grades:[6, 6]},{name:"Donald",age:10,grades:[]},{name:"Isacc",age:20,grades:[2, 2, 3]},{name:"Alex",age:11,grades:[6, 6, 3]}] | Johnny : 11 -> 6, 4, 3  Peter : 10 -> 5, 5, 4  Jordan : 13 -> 6, 6  Donald : 10 -> None  Isacc : 20 -> 2, 2, 3  Alex : 11 -> 6, 6, 3  stringify |

## Sentence Split

Splitting the input by a given delimiter is easy, especially when you are told in the problem descriptions, that the elements will not contain the delimiter. What if, however, they do.

You will be **given a sentence**, which may contain **ANY ASCII CHARACTER**.  
Then you will be **given** a **delimiter**, which may **ALSO** contain **ANY ASCII CHARACTER**.

Your task is to **split the sentence**, by the **given** **delimiter**, and print each of the elements, in the following format:

[{element1}, {element2}, {element3}. . .]

See the examples for more info.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| This will be hard, right. Nope!  . **(dot and space)** | [This will be hard, right, Nope!] |
| Bau, Chika, Bow Wow  , **(comma and space)** | [Bau, Chika, Bow Wow] |

## Capitalize Words

Create a program, which receives sentences of words, and capitalizes the words in them.

The sentences will follow a **correct grammatical order**. You must **extract** **each** **word** from the sentences, and you must **capitalize** it.

The **capitalization** of a **word**, is the process of making the **first letter** from it – **Capital**, and **every other** – **lowercase**.

You should read input lines, until you receive the command “**end**”.

You **might** receive **SEVERAL sentences** on **1 input line**.

After you’ve read a sentence, you extract the words, **capitalize them**, and **print them**, **separated** by a **space** and a **comma**. **ONLY** then you should **read** the **next sentence**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| This is funny.  This is not.  end | This, Is, Funny  This, Is, Not |
| Let’s... gEt... iT RoCKing In HeRE!!!  PartY rock ANTHEM! ! !  end | Let’s, Get, It, Rocking, In, Here  Party, Rock, Anthem |