# More Exercises: String and Text Processing

Problems for in-class lab for the: [“Technology Fundamentals” course @ SoftUni](https://softuni.bg/courses/technology-fundamentals).

Submit your solutions in the SoftUni judge system at**:** [Strings-and-Text-Processing-More-Exercise](https://judge.softuni.bg/Contests/1339/Strings-and-Text-Processing-More-Exercises)

## Find substrings

Write a JS Function to **find how many times a given string appears in a given text as substring**. The input will be **two parameters: two strings**. The first string will be the **text**, and the second will be the **searched substring**. The output is an integer number. Please ignore the **character casing**. **Overlapping** between occurrences is **allowed**.

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| **"Wel**come to the Software University (SoftUni)! **Wel**come to programming. Programming is **wel**lness for developers, said Max**wel**l.", "wel" | 4 |

## Substring extractor

Find and remove **all words** in a text that **contain given substring (case insensitive)**. The input will come as **two strings: the searched substring and the text to search in**. Then print the remaining words with their indices in the following format: **"idx[{index}] -> {currentWord}"**

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| 'wel','Welcome to the Software University!' | idx[0] -> to  idx[1] -> the  idx[2] -> Software  idx[3] -> University! |

## Santa's reindeer

Santa has lost some of his reindeer the night before X-Mas. You are tasked to find them in a mountain of text and save X-Mas. You need to call them by their names and you will find each reindeer surrounded by snowflakes ("\*"). Reindeer's name always starts with capital letter and is followed only by small letters. Extract the reindeers from the text. Print them joined by **", "**, then print the remaining text with a **separating line of 20 snowflakes**.

### Constrains

* **Solve this problem without Regex!**

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| "Oh my \*Rudolf\* dwarfs! This \*Dasher\* year's christmas \*Prancer is about to be Comet\* ruined because Santa has lost his deer and." | \*Rudolf\*, \*Dasher\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Oh my dwarfs! This year's christmas \*Prancer is about to be Comet\* ruined because Santa has lost his deer and. |

## Snow white

After Snow white was poisoned by the apple she ate, she started forgetting her dwarf's names. Help her remember each dwarf name by finding the keyword ("dwarf") and placing each dwarf's name right after it. If you run out of names just remove the keyword.

As input, you will be given **2 parameters** (text and names). Text will be **a string** and names will be **array of strings**.

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| "Yet another dwarf fairytale tragedy! There are dwarf so many dwarfs, is it dwarf possible to help dwarf Show white?",  ["Doc", "Dopey", "Sleepy"] | Yet another dwarf Doc fairytale tragedy! There are dwarf Dopey so many dwarfs, is it dwarf Sleepy possible to help Show white? |

## 5. Date Finder

Write a JS function that finds and extracts all the dates in the given sentences. The dates should be in format   
**d-MMM-yyyy**. **Example: 12-Jun-1999**, **3-Dec-2017**.

The **input** comes as an **array of strings**. Each string represents a sentence.

The **output** should be printed on the console. The output should consist of all extracted **VALID** dates. Each element should be printed on a new line.

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['I am born on 28-Feb-1994.','This is not date: 512-Jan-1996.','My father is born on the 29-Jul-1955.'] | 28 Feb 1994   1. ul 1955 |

## Wall builder

In this problem we'll build BIG WALLS, but our material is characters from the ASCII table.

A strong wall is created in three stages:

**First stage is the basis**.  
Since it is the basis, it must be SOLID. This means that it contains only **capital letters** and at least **four-digit numbers.** The letters and numbers must alternate one after the other. Basis stage always starts with letter and ends with number.

**Example:** SO9000STRONG10000WALL7000

**The second stage is the middle one.** It can contain **uppercase and lowercase letters and special characters** like these: **#, @, $ and %.**   
Base string always **starts** with at least **3 upper-case and / or lower-case** letters and **ends** **with one of the special characters.**

**Example:** xaXAxa###MiDDlE%%$##

**The last stage** of our wall is called **top**:   
The top must be light weight, that's why **we have only zeros and lower-case letters**, but of course we must follow the rules to get solid structure. Top string **always starts with zeros and ends with lower-case letters.** A single top **should not have more than 5 consecutive letters.**

**Example:** 00000cant0do000that

Тhe array length will always be divisible to 3 without a remainder.  
So you already know if you receive array with 9 elements the first 3 of them are Bases, next 3 are Middles and the last 3 are Tops

If some stage is solid, print **'SOLID {stage}!'**

If it's not, print **'WEAK {stage}!'**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['SO9000STRONG10000WALL7000', 'YES345345345PLEASEHAHA999999',  'xaXAxa###MiDDlE%%$##', 'mMmMm%%%%ImRichGuy$$$$$',  '00000cant0do000that', '0sorry00happy000sad'] | SOLID BASE!  SOLID BASE!  SOLID MIDDLE!  SOLID MIDDLE!  SOLID TOP!  SOLID TOP! |
| [  'SO9000STRONG10WALL7000', 'YES345345345PLEASEHAHA999999',  'xaXAxa###MiDDlE%%$##', 'mM%%%%ImRichGuy$$$$$',  '00000cant0do000that', '0sorry00happy000saaaad'  ] | WEAK BASE!  SOLID BASE!  SOLID MIDDLE!  WEAK MIDDLE!  SOLID TOP!  WEAK TOP! |