Exercises: Files, Directories and Exceptions

Problems for exercises and homework for the "Programming Fundamentals" course @ SoftUni.

This exercise does **NOT** have **Judge Contest**. That means that you will need to **create input and output files** from the examples and **test** the solutions on your own.

1. Most Frequent Number

Write a program that finds the **most frequent number** in a given sequence of numbers.

- Numbers will be in the range **[0…65535]**.
- In case of multiple numbers with the same maximum frequency, print the **leftmost** one.

Read the input from a file and save the result in new file "output.txt"

Examples

input.txt	output.txt	Comments
4 1 1 **4** 2 3 **4** **4** 1 2 **4** 9 3 **2** **2** **2** **2** 1 **2** **2** **2** **7** **7** **7** 0 2 2 2 0 10 10 10	4 2 7	The number **4** is the most frequent (occurs 5 times) The number **2** is the most frequent (occurs 7 times) The numbers **2**, **7** and **10** have the same maximal frequence. The leftmost of them is **7**.

2. Index of Letters

Write a program that creates an array containing all letters from the alphabet (**a**-**z**). Read a lowercase word from a file "input.txt" and write the **index of each of its letters in the letters array in the "output.txt" file**.

Examples

input.txt	output.txt
abcz	a -> 0 b -> 1 c -> 2 z -> 25
softuni	s -> 18 o -> 14 f -> 5 t -> 19 u -> 20 n -> 13 i -> 8

3. Count of Symbols

Write a program that reads a text from a file "input.txt" and counts the occurrences of each symbol in the text. Write in a file each symbol along its count, ordered by most occurrences.

Examples

input.txt	output.txt
Hello, C#!	l -> 2 H -> 1 e -> 1 o -> 1 , -> 1 C -> 1 # -> 1 ! -> 1

4. Max Sequence of Equal Elements

Read a **file with lines of integers** and find the **longest sequence of equal elements on each line**. If several exist, take the **leftmost**. Write the result in a file output.txt.

Examples

input.txt	output.txt
3 4 4 **5 5 5** 2 2 **7 7** 4 4 5 5 3 3 1 2 **3 3**	5 5 5 7 7 3 3

Hints

- Scan positions **p** from left to right and keep the **start** and **length** of the current sequence of equal numbers ending at **p**.
- Keep also the currently best (longest) sequence (**`bestStart`** + **`bestLength`**) and update it after each step.

5. Fix Emails

You are given a sequence of strings, each on a new line, **until you receive "stop" command**. First string is a name of a person. On the second line, you receive his email. Your task is to collect their names and emails, and remove emails whose domain ends with "**us**" or "**uk**" (case insensitive). Print:

{name} – > {email}

Examples

input.txt	output.txt
Ivan ivanivan@abv.bg Petar Ivanov petartudjarov@abv.bg Mike Tyson myke@gmail.us stop	Ivan -> ivanivan@abv.bg Petar Ivanov -> petartudjarov@abv.bg

6. Advertisement Message

Write a program that **generate random fake advertisement message** to extol some product. The messages must consist of 4 parts: laudatory **phrase** + **event** + **author** + **city**. Use the following predefined parts:

- **Phrases** – {"Excellent product.", "Such a great product.", "I always use that product.", "Best product of its category.", "Exceptional product.", "I can't live without this product."}
- **Events** – {"Now I feel good.", "I have succeeded with this product.", "Makes miracles. I am happy of the results!", "I cannot believe but now I feel awesome.", "Try it yourself, I am very satisfied.", "I feel great!"}
- **Author** – {"Diana", "Petya", "Stella", "Elena", "Katya", "Iva", "Annie", "Eva"}
- **Cities** – {"Burgas", "Sofia", "Plovdiv", "Varna", "Ruse"}

The format of the output message is: **`{phrase} {event} {author} - {city}`**.

As an input, you take the **number of messages from the console** to be generated. Write each random message at a separate line in the output.txt.

Examples

Input	output.txt
3	Such a great product. Now I feel good. Elena – Ruse Excelent product. Makes miracles. I am happy of the results! Katya – Varna Best product of its category. That makes miracles. Eva - Sofia

Hints

- Hold the **`phrases`**, **`events`**, **`authors`** and **`towns`** in 4 arrays of strings.
- Create **`Random`** object and generate 4 random numbers each in its range:
  - `phraseIndex → [0, phrases.Length]`
  - `eventIndex → [0, events.Length]`
  - `authorIndex → [0, authors.Length]`
  - `townIndex → [0, towns.Length]`
- Get one **random element** from each of the four arrays and **compose a message** in the required format.