Exercises: Lists

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

Submit your solutions in the SoftUni judge system at https://judge.softuni.bg/Contests/1087/Lists-Exercises.

01. Sum Adjacent Equal Numbers

Write a program to **sum all adjacent equal numbers** in a list of decimal numbers, starting from **left to right**.

- After two numbers are summed, the obtained result could be equal to some of its neighbors and should be summed as well (see the examples below).
- Always sum the **leftmost** two equal neighbors (if several couples of equal neighbors are available).

Examples

Input	Output	Explanation
3 3 6 1	12 1	**3 3** 6 1 → **6 6** 1 → 12 1
8 2 2 4 8 16	16 8 16	8 **2 2** 4 8 16 → 8 **4 4** 8 16 → **8 8** 8 16 → 16 8 16
5 4 2 1 1 4	5 8 4	5 4 2 **1 1** 4 → 5 4 **2 2** 4 → 5 **4 4** 4 → 5 8 4

Hints

1. Read the **input** and parse it to **list of numbers**.
2. Find the **leftmost** two **adjacent equal cells**.
3. **Replace** them with their **sum**.
4. **Repeat** (1) and (2) until no two equal adjacent cells survive.
5. **Print** the processed list of numbers.

02. Split by Word Casing

Read a **text**, split it into words and distribute them into **3 lists**.

- **Lower-case words** like "programming", "at" and "databases" – consist of lowercase letters only.
- **Upper-case words** like "PHP", "JS" and "SQL" – consist of uppercase letters only.
- **Mixed-case words** like "C#", "SoftUni" and "Java" – all others.

Use the following **separators** between the words: **`, ; : . ! ( ) " ' \ / [ ] space`**

Print the 3 lists as shown in the example below.

Examples

Input	Output

`Learn programming at SoftUni: Java, PHP, JS, HTML 5, CSS, Web, C#, SQL, databases, AJAX, etc.`	`Lower-case: programming, at, databases, etc`
	`Mixed-case: Learn, SoftUni, Java, 5, Web, C#`
	`Upper-case: PHP, JS, HTML, CSS, SQL, AJAX`

Hints

- **Split** the input text using the above described **separators**.

- **Process** the obtained **list of words** one by one.
- Create 3 lists of words: lowercase words, mixed-case words and uppercase words.
- Check each word and append it to one of the above 3 lists:
  - Count the **lowercase letters** and **uppercase letters**.
  - If all letters are **lowercase**, append the word to the lowercase list.
  - If all letters are **uppercase**, append the word to the uppercase list.
  - Otherwise the word is considered mixed-case → append it to the mixed-case list.
- Print the obtained 3 lists as shown in the example above.

03. Count Numbers

Read a **list of integers** in range [0…1000] and **print them in ascending order** along with their **number of occurrences**.

Examples

Input	Output
8 2 2 8 2 2 3 7	2 -> 4 3 -> 1 7 -> 1 8 -> 2
10 8 8 10 10	8 -> 2 10 -> 3

Hints

Several algorithms can solve this problem:

- Use an **list `count`** to count in **`counts[x]`** the occurrences of each item **`x`**.
- **Sort** the numbers and count occurrences of each number.

Counting Occurrences Using List

1. Read the input items in list of integers.
2. Allocate a list **`counts`**.
   - It will hold for each number **`x`** its number of occurrences **`counts[x]`**.
3. **Scan** the input items and for each item **`x`** increase **`counts[x]`**.

This algorithm has a **serious drawback**:

- It depends on **mapping numbers to list indexes**.
- It will work well for input values in the range [0…1000].
- It will **not work** for very large and very small values, e.g. if the input holds -100 000 000 or 100 000 000.
- It will **not work** for real numbers, e.g. 3.14 or 2.5.

Counting Occurrences by After Sorting

1. Read the input items in list of integers. Example: {8, 2, 2, 8, 2, 2, 3, 7}.
2. Sort **`the list`** in increasing order: {2, 2, 2, 2, 3, 7, 8, 8}. Now find all subsequences of equal numbers.
3. **Scan** the numbers from left to right. Count how many times each number occurs.
   - Start at **`count`** = **`1`**.

- While the next number on the right is **the same** as the current number, **increase `count`** and proceed to the next number.
- When the next number on the right is **different** (or there is no next number), **print** the current number and its count.
- Continue scanning from the next number on the right.

This algorithm will work correctly for real numbers and very large numbers. It does not depend on mapping numbers to list indexes.

Note: for this exercise, you are not allowed to use ready functions for doing the algorithm. They are allowed only for reading the input and printing the result.

04. List Contains Item

Read a **list of integers** on the first line of the console and an integer **N** from the second line of the console and print whether the item is **contained** in the list. If it is, print "**yes**", otherwise print "**no**".

Examples

Input	Output
1 2 3 4 5 5	yes
8 7 7 9 6 2 2 11	no
99 7 8 6 2314 2 2314	yes

Hints

- Read a text line from the console, split it by space, parse the obtained items as integers and convert them to list of integers.
- Scan through the whole list, item by item, until you either find the item, or reach the end of the list.
- Keep the result of the operation in a Boolean variable such as "**`isFound`**".
- Finally, if the item is found (checking by the variable), print "**yes**" or "**no**".

05. Smallest Element in List

Read a **list of integers** on the first line of the console. After that, **find** the smallest item in the list and **print** it on the console.

Examples

Input	Output
1 2 3 4 5	1
9 8 7 82 78 13	7
78 77 1268 43 9	9

Hints

- Read a text line from the console, split it by space, parse the obtained items as integers and convert them to list of integers.
- Traverse the whole list, item by item, putting the **smallest integer** up to that point into an integer variable called "**`smallestInt`**". Finally, print **`smallestInt`**.

06. Reverse List In-place

Read a **list of integers** on the first line of the console. After that, **reverse** the list in-place (as in, don't create a new collection to hold the result, reverse it using only the original list). After you are done, **print** the reversed list on the console.

Note: You are **not** allowed to iterate over the list backwards and just print it,

Examples

Input	Output
1 2 3 4 5	5 4 3 2 1
1 4 2 7 6 1 1	1 1 6 7 2 4 1
11 52 43 12 1 6	6 1 12 43 52 11

Hints

- Iterate over **half** of the list (0…length / 2) and swap each item with its opposite index like so:
  - Index 0 gets swapped with -1
  - Index 1 gets swapped with -2,
  - Index 2 gets swapped with -3,
  - and so on…
- When you reach the **middle** of the list, it means you are done swapping items and are ready to print them.

07. Sort List Using Bubble Sort*

Read a **list of integers** on the first line of the console. After that, **sort** the list, using the **Bubble Sort** algorithm.

Examples

Input	Output
5 3 4 1 2	1 2 3 4 5
11 872 673 1 2	1 2 11 673 872
11 52 43 12 1 6	1 6 11 12 43 52

08. Sort List Using Insertion Sort*

Read a **list of integers** on the first line of the console. After that, **sort** the list, using the **Insertion Sort** algorithm.

Examples

Input	Output
5 3 4 1 2	1 2 3 4 5

11 872 673 1 2	1 2 11 673 872
11 52 43 12 1 6	1 6 11 12 43 52

09. Largest N Items*

Read a **list of integers** on the first line of the console. On the next line, you will receive an **integer N**. After that, find and **print** the **largest N items** the list, sorted in **descending order**.

Examples

Input	Output
5 3 4 1 2 3	5 4 3
11 872 673 1 2 2	872 673
11 52 43 12 1 6 4	52 43 12 11

Hints

- A possible solution to this problem is:
  - Sort the list in **descending order**, using a sorting algorithm such as **Insertion Sort**,
  - Extract the first **N items** from it