# Exercises: Lists

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

You can check your solutions here: <https://judge.softuni.bg/Contests/424>.

## Remove Elements at Odd Positions

Write a program which reads a **list** of **space-separated** **strings** and **removes** the elements at positions [1…3…5…7…etc.]. After that, **print** the elements with **no delimiter**. Note that positions are **counted from 1**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| First Soft ccc Uni fifth | SoftUni |
| now you're\_ fired hired | you're\_hired |
| java I\_ x < php 3\_ - C# | I\_<3\_C# |

### Hints

* Instead of removing elements from the initial list, **create** a new list and **add** only elements which **aren’t contained** in the new list yet.

## Track Downloader

You will receive a list of **blacklisted words** (on the same line, separated by **one** space). On the next lines, you will start receiving a list of **filenames** (as **strings**) until you receive the string “end”. Your task is to **add** the filenames to a **list** and **sort** them. **Ignore** the tracks which **contain** the text in the **blacklist**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Metallica Linkin Park  01. Ceca - Maskarada.mp3  6.Metallica–Nothing Else Matters.mp3  Kilo\_dole\_kilo\_gore\_superfreedotbiz.mp3  Metallica\_forWhomTheBellTolls.flac  04. ...Powdered Water Too (Part 1).mp3  Linkin Park - Numb.mp3  10. One Word Extinguisher.flac  end | 01. Ceca - Maskarada.mp3  04. ...Powdered Water Too (Part 1).mp3  10. One Word Extinguisher.flac  Kilo\_dole\_kilo\_gore\_superfreedotbiz.mp3 |
| Britney  12 - The Messenger.mp3  01-Can't Knock The Hustle.mp3  09-99 Problems.mp3  Britney Spears - Toxic.mp3  03 - Axion.mp3  07 - Ultra Thizz.flac  Britney - Baby one more time.flac  03 - Surph.flac  02 - Breakbeat Malaria.mp3  end | 01-Can't Knock The Hustle.mp3  02 - Breakbeat Malaria.mp3  03 - Axion.mp3  03 - Surph.flac  07 - Ultra Thizz.flac  09-99 Problems.mp3  12 - The Messenger.mp3 |
| -  02 - Guillotine.mp3  Death Grips - Takyon (Death Yon).mp3  03. Started From The Bottom.mp3  Tuscan\_Leather\_V0.mp3  10. Big Shots.mp3  11 - Void (Internal Theory).mp3  01. Jynweythek.flac  end | 01. Jynweythek.flac  03. Started From The Bottom.mp3  10. Big Shots.mp3  Tuscan\_Leather\_V0.mp3 |

### Hints

* You can check if a string contains another string by using the string.Contains(str) method.

## Equal Sum After Extraction

You will be given **two** **integer** **lists** on the first **two lines** (space-separated). **Remove** the elements in the **first list** from the elements in the **second list**. If an element from the first list occurs **more than once** in the second list, **remove all occurrences**.

After you remove the elements, check the **sum** of both lists. If the sum of both of them is **equal**, print “Yes. Sum: {sum}”. If not print the **absolute** difference between the two lists in the format “No. Diff: {sum}”.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 1 2 3 4 5  5 3 2 7 4 1 8 | Yes. Sum: 15 | first list: 1 2 3 4 5  second list: 5 3 2 7 4 1 8  elements to remove: 5 3 2 7 4 1 8  resulting list: 7 8  list1 sum: 1 + 2 + 3 + 4 + 5 = 15  list2 sum: 7 + 8 = 15  list1 sum == list2 sum 🡺 Equal sum |
| 3 4 4 6 7  20 2 2 | Yes. Sum: 24 | no list1 elements present in list2 🡺 no removal  list1 sum = 3 + 4 = 4 + 6 + 7 = 24  list2 sum = 20 + 2 + 2 = 24 |
| 8 7 9 8 7  1 2 3 8 4 | No. Diff: 29 | elements to remove: 1 2 3 8 4  list1 sum = 8 + 7 + 9 + 8 + 7 = 39  list2 sum = 1 + 2 + 3 + 4 = 10  list1 sum != list2 sum 🡺 Diff: 39 - 10 = 29 |

## Flip List Sides

You will receive an integer list on the first line of the input (space-separated). Leave the first, middle and last elements as they are. For every other element, **exchange** it with its **opposite indexed element** (list[1] 🡺 list[length-2] and so on…). After that, print the list on the console (space-separated).

|  |  |  |
| --- | --- | --- |
|  | 🡺 |  |

*Note: If the list has an even number of elements, as there is no middle element,* ***flip*** *all the elements* ***except*** *for the* ***first*** *and* ***last*** *ones.*

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 2 3 4 5 6 7 8 9 10 11 | 1 10 9 8 7 6 5 4 3 2 11 |
| 2 3 4 1 | 2 4 3 1 |
| 12 88 9 7 7623 84 2 1 8 97 | 12 8 1 2 84 7623 7 9 88 97 |

## \* Tear List in Half

You will receive a list of **integers** on the **first** input line (space-separated). Your task is to separate the list into **two** parts and perform the following operations:

|  |
| --- |
| Assume you have the following initial list: |
|  |
| Split the list into **2 equal parts** and stack the **right half** on top of the **left half**: |
|  |
| Place **each digit** of the upper list elements on **either side** of the lower list: |
|  |

After you perform the operations, print the resulting list on the console.

### Constraints

The count of the input list elements will **always** be **even** and the elements will always be **between 10 and 99**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 11 22 33 45 14 12 | 4 11 5 1 22 4 1 33 2 |
| 82 77 12 84 92 11 | 8 82 4 9 77 2 1 12 1 |
| 11 12 13 14 15 16 17 18 | 1 11 5 1 12 6 1 13 7 1 14 8 |

### Hints

* There is a certain arithmetic progression which can help you solve this problem:
  + Simulate the *splitting list -> splitting numbers into digits -> inserting to the left and right of the element* action several times and pay attention to the indices you are inserting the numbers into.
  + Do this several times and you will find the progression by yourself.
  + It is the key to seeing at which index you need to insert the elements each time and it works no matter how many elements the list has as long as you follow it.

## \* Stuck Zipper

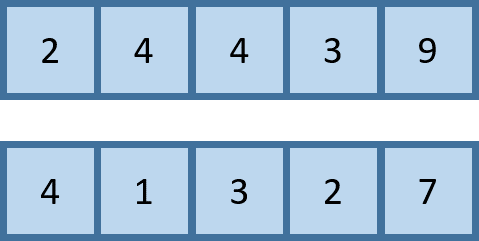
You will be given **two** lists of **integers** on the first and second line of the input respectively (space-separated). We want to **zip** these two lists together like a **zipper**, but **one or several** of the elements in **either** of the lists are **too large** to allow that to happen. Being larger, they have **more digits** than all the other elements in the two lists.

Your task is to find any elements which have **more digits** than the element with the least amount of digits in the list (**if such exist**) and **remove** them, then **zip** the lists together as shown in the example.

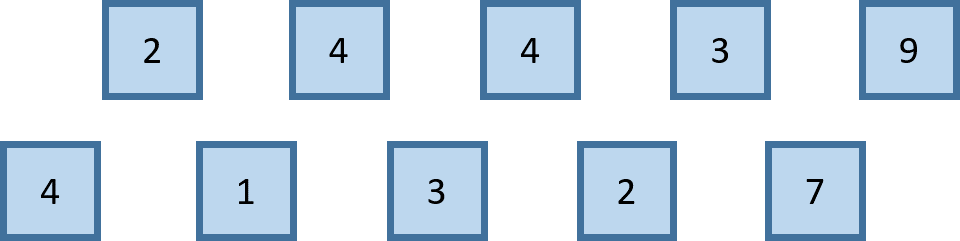
After you stack the lists, **print** the resulting list on the console (one line, space-separated).

Ideal example (no bad elements):

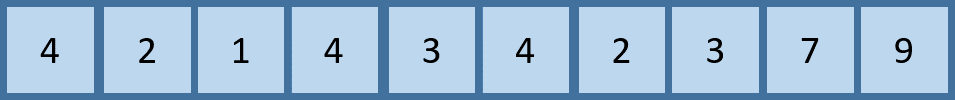
The initial lists:



Separate the elements to prepare for zipping:

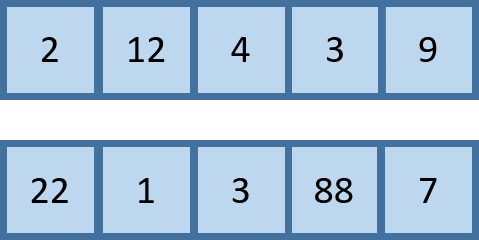


Zip the list:

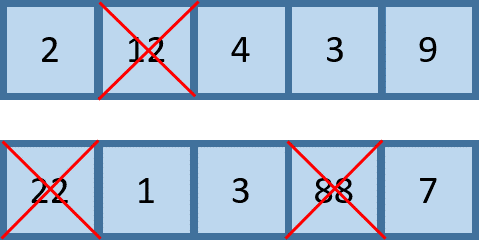


Stuck zipper example:

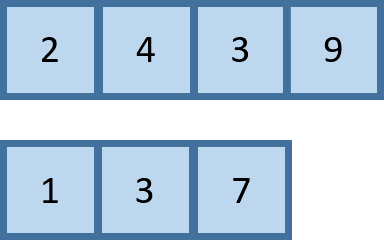
The initial lists:



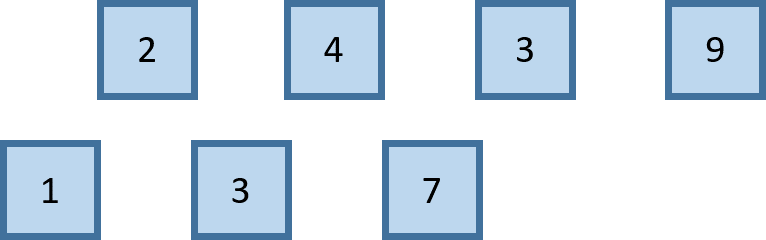
Remove the elements with **more digits** than the element with the **smallest amount** **of digits** in both of the lists. In this case, we have to remove anything with more than **1** digit:



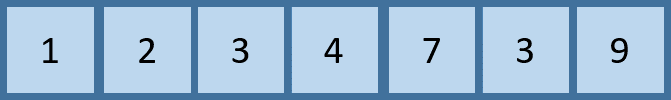
Lists after removal:



Separate the elements to prepare for zipping:



Finally, we zip the list:



### Constraints

The integers in both of the lists will always be in the range [1…1000000]

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 2 4 4 3 9  4 1 3 2 7 | 4 2 1 4 3 4 2 3 7 9 | Nothing to remove – just zip the lists.  Insert list1[0] into list2 at index 1,  list1[1] into list2 at index 3,  list1[2] into list2 at index 5, etc... |
| 2 12 4 3 9  22 1 3 88 7 | 1 2 3 4 7 3 9 | We find any elements which have more than **1 digits** in both lists. We then zip the two lists like so:  Insert list1[0] into list2 at index 1,  list1[1] into list2 at index 3,  list1[2] into list2 at index 5, etc... |
| -156 9128 7223 -938 -923  -918 22282 -848 98127 | -918 -156 -848 -938 -923 | We find any elements which have more than **3 digits** and **remove** them. We then zip the two lists. |
| 81728 73234 89 9912  2 5379 -5 92348 | 2 -5 | Find elements which have more than **1 digit** and **remove** them.  Since we have nothing to zip (first list is empty), we just print what’s left. |