# Lab: Lists

Problems for in-class lab for the [["C# Fundamentals" course @ SoftUni](https://softuni.bg/trainings/2363/csharp-fundamentals-may-2019)](https://softuni.bg/modules/57/tech-module-4-0)  
You can check your solutions in [Judge](https://judge.softuni.bg/Contests/1210)

## 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 |

## Gauss' Trick

Write a program that **sums** all of the **numbers in a list** in the following order:

*first* ***+*** *last*, *first + 1* ***+*** *last - 1*,*first + 2* **+** *last* ***-*** *2***,** … *first + n*, *last - n*.



### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 2 3 4 5 | 6 6 3 |
| 1 2 3 4 | 5 5 |

## Merging Lists

You are going to receive two lists with numbers. Create a result list, which contains the numbers from both of the lists. The **first** **element** should be **from the first list**, the **second** from the **second list** and so on. If the length of the two lists **are not equal**, just **add** the **remaining** **elements** at the **end** of the list.

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| **3 5 2 43 12 3 54 10 23**  **76 5 34 2 4 12** | **3** **76** **5** **5** **2** **34** **43** **2** **12** **4** **3** **12** **54** **10** **23** |
| **76 5 34 2 4 12**  **3 5 2 43 12 3 54 10 23** | **76** **3** 5 **5** **34** **2** **2** **43** **4** **12** **12** **3** **54 10 23** |

## List of Products

Read a number **n** and **n lines of products**. Print a **numbered list** of all the products **ordered by name**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4  Potatoes  Tomatoes  Onions  Apples | 1.Apples  2.Onions  3.Potatoes  4.Tomatoes |

## Remove Negatives and Reverse

Read a **list of integers**, **remove all negative numbers** from it and print the remaining elements in **reversed order**. In case there are no elements left in the list, print "empty".

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 10 -5 7 9 -33 50 | 50 9 7 10 |
| 7 -2 -10 1 | 1 7 |
| -1 -2 -3 | empty |

## List Manipulation Basics

Write a program that reads a list of integers. Then until you receive **"end"**, you will receive different **commands:**

**Add {number}:** add a number to the end of the list.  
**Remove {number}:** remove a number from the list.  
**RemoveAt {index}:** remove a number at a given index.  
**Insert {number} {index}:** insert a number at a given index.

**Note: All the indices will be valid!**

When you receive the **"end"** command, print the **final state** of the list (**separated by spaces**).

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4 19 2 53 6 43  Add 3  Remove 2  RemoveAt 1  Insert 8 3  end | 4 53 6 8 43 3 |

## List Manipulation Advanced

Next, we are going to implement more complicated list commands, **extending the previous task**. Again, read a list and keep reading commands until you receive **"end"**:

**Contains {number}** – check if the list contains the number and if so - print **"Yes**", **otherwise** print **"No such number"**.  
**PrintEven** – print **all the even numbers**, **separated by a space**.  
**PrintOdd** – print **all the odd numbers, separated by a space**. **GetSum** – print the **sum of all the numbers**. **Filter {condition} {number}** – print all the numbers that **fulfill the given condition**. The condition will be either '**<**', '**>**', "**>=**","**<=**".

**After** the **end** **command,** print the list **only if** you have made some **changes** to the **original list**. **Changes** are made **only** from the commands from the **previous task**.

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 13 43 876 342 23 543  Contains 100  Contains 543  PrintEven  PrintOdd  GetSum  Filter >= 43  Filter < 100  end | No such number  Yes  2 876 342  13 43 23 543  1842  43 876 342 543  2 13 43 23 |