

You can check your solutions in [Judge](#)

Lab: Functional Programming.....	1
1. Sort Even Numbers.....	1
2. Sum Numbers	1
3. Count Uppercase Words	2
4. Add VAT.....	2
5. Filter by Age.....	2
Exercises: Functional Programming.....	3
1. Action Print.....	3
2. Knights of Honor	3
3. Custom Min Function	4
4. Find Evens or Odds.....	4
5. Applied Arithmetics.....	4
6. Reverse and Exclude.....	5
7. Predicate for Names.....	5
8. List of Predicates	5
9. Predicate Party!.....	6
10. Party Reservation Filter Module	6
11. TriFunction	7

1. SORT EVEN NUMBERS

Create a program that reads one line of **integers** separated by ", ". Then prints the **even numbers** of that sequence **sorted** in **increasing** order.

EXAMPLES

Input	Output	Input	Output	Input	Output
4, 2, 1, 3, 5, 7, 1, 4, 2, 12	2, 2, 4, 4, 12	1, 3, 5		2, 4, 6	2, 4, 6

HINT

It is up to you what type of data structures you will use to solve this problem. Use a functional programming filter and sort the collection of numbers.

2. SUM NUMBERS

Create a program that reads a line of **integers** separated by ", ". Print on two lines the **count** of numbers and their **sum**.

EXAMPLES

Input	Output
-------	--------

4, 2, 1, 3, 5, 7, 1, 4, 2, 12	10 41
2, 4, 6	3 12

3. COUNT UPPERCASE WORDS

Create a program that reads a line of **text** from the console. Print **all** the words that start with an **uppercase letter** in the **same order** you've received them in the text.

EXAMPLES

Input	Output
The following example shows how to use Function	The Function
Write a program that reads one line of text from console. Print count of words that start with Uppercase, after that print all those words in the same order like you find them in text.	Write Print Uppercase,

HINT

Use **Func<string, bool>** and use ' ' for splitting words.

4. ADD VAT

Create a program that reads one line of **double** prices separated by ", ". Print the **prices** with **added VAT** for all of them. **Format** them to **2 signs** after the decimal point. The **order** of the prices must be the **same**. **VAT** is equal to **20%** of the price.

EXAMPLES

Input	Output	Input	Output
1.38, 2.56, 4.4	1.66 3.07 5.28	1, 3, 5, 7	1.20 3.60 6.00 8.40

5. FILTER BY AGE

Write a program that receives an integer **N** on the first line. On the next **N** lines, read pairs of "[name], [age]". Then read three lines:

- **Condition** - "younger" (<) or "older" (>=)
- **Age threshold** - integer
- **Format** - "name", "age" or "name age"

Depending on the **condition**, print the correct **pairs** in the correct **format**. **Don't use the built-in functionality from .NET**. Create your own methods.

EXAMPLES

Input	Output	Input	Output	Input	Output
-------	--------	-------	--------	-------	--------

5 Lucas, 20 Tomas, 18 Mia, 29 Noah, 31 Simo, 16 older 20 name age	Lucas - 20 Mia - 29 Noah - 31	5 Lucas, 20 Tomas, 18 Mia, 29 Noah, 31 Simo, 16 younger 20 name	Tomas Simo	5 Lucas, 20 Tomas, 18 Mia, 29 Noah, 31 Simo, 16 younger 50 age	20 18 29 31 16
---	-------------------------------------	---	---------------	--	----------------------------

HINTS

Implement the following steps:

```
List<Person> people = ReadPeople();
Func<Person, bool> filter = CreateFilter(condition, ageThreshold);
Action<Person> printer = CreatePrinter(format);
PrintFilteredPeople(people, filter, printer);
```

The methods **CreateFilter(condition, ageThreshold)** and **CreatePrinter(format)** should return **lambda functions** as output.

EXERCISES: FUNCTIONAL PROGRAMMING

- You can check your solutions in [Judge](#)
- Ask your questions here <https://www.slido.com/> by entering the code **#csharp-advanced**

1. ACTION PRINT

Create a program that reads a collection of **strings** from the console and then **prints** them onto the **console**. Each name should be printed on a **new line**. Use **Action<T>**.

EXAMPLES

Input	Output
Lucas Noah Tea	Lucas Noah Tea
Teo Lucas Harry	Teo Lucas Harry
Ashurbanipal Napoleon Caesar	Ashurbanipal Napoleon Caesar

2. KNIGHTS OF HONOR

Create a program that reads a collection of **names** as **strings** from the **console**, appends **"Sir"** in front of every name and **prints** it back in the **console**. Use **Action<T>**.

EXAMPLES

Input	Output
Eathan Lucas Noah Arthur	Sir Eathan Sir Lucas Sir Noah Sir Arthur
Lucas Jade Hugo	Sir Lucas Sir Jade Sir Hugo
Ashurbanipal Napoleon Caesar	Sir Ashurbanipal Sir Napoleon Sir Caesar

3. CUSTOM MIN FUNCTION

Create a simple program that reads from the **console** a set of **integers** and **prints** back on the **console** the **smallest number** from the collection. Use **Func<T, T>**.

EXAMPLES

Input	Output
1 4 3 2 1 7 13	1
4 5 -2 3 -5 8	-5

4. FIND EVENS OR ODDS

You are given a lower and an upper bound for a range of integer numbers. Then a command specifies if you need to list all even or odd numbers in the given range. Use **Predicate<T>**.

EXAMPLES

Input	Output
1 10 odd	1 3 5 7 9
20 30 even	20 22 24 26 28 30

5. APPLIED ARITHMETICS

Create a program that executes some mathematical operations on a given collection. On the **first line**, you are given a **list of numbers**. On the **next lines**, you are passed **different commands** that you need to **apply to all the numbers** in the list:

- **"add"** -> add 1 to each number
- **"multiply"** -> multiply each number by 2
- **"subtract"** -> subtract 1 from each number
- **"print"** -> print the collection
- **"end"** -> ends the input

Note: Use functions.

EXAMPLES

Input	Output
1 2 3 4 5 add add print end	3 4 5 6 7
5 10 multiply subtract print end	9 19

6. REVERSE AND EXCLUDE

Create a program that reverses a collection and removes elements that are divisible by a given integer **n**. Use predicates / functions.

EXAMPLES

Input	Output
1 2 3 4 5 6 2	5 3 1
20 10 40 30 60 50 3	50 40 10 20

7. PREDICATE FOR NAMES

Write a program that filters a list of names according to their length. On the first line, you will be given an integer **n**, representing a name's length. On the second line, you will be given some names as strings separated by space. Write a function that prints only the names whose length is **less than or equal** to **n**.

EXAMPLES

Input	Output
4 Karl Anna Kris Yahto	Karl Anna Kris
4 Karl James George Robert Patricia	Karl

8. LIST OF PREDICATES

Find all numbers in the range **1...N** that are divisible by the numbers of a given sequence. On the first line, you will be given an integer **N** – which is the end of the range. On the second line, you will be given a sequence of integers which are the **dividers**. Use **predicates / functions**.

EXAMPLES

Input	Output
10 1 1 1 2	2 4 6 8 10
100 2 5 10 20	20 40 60 80 100

9. PREDICATE PARTY!



Ivan's parents are on a vacation for the holidays and he is planning an epic party at home. Unfortunately, his organizational skills are next to non-existent, so you are given the task to help him with the reservations.

On the **first line**, you receive a **list of all the people** that are coming. On the **next lines**, until you get the **"Party!" command**, you may be asked to **double** or **remove all the people** that apply to the given **criteria**. There are **three different criteria**:

- Everyone that has a **name starting** with a **given string**
- Everyone that has a **name ending** with a **given string**
- Everyone that has a **name** with a **given length**

Finally, **print all the guests** who are going to the party **separated by ", "** and then **add the ending "are going to the party!"**. If no guests are going to the party print **"Nobody is going to the party!"**. See the examples below:

EXAMPLES

Input	Output
Peter Misha Stephen Remove StartsWith P Double Length 5 Party!	Misha, Misha, Stephen are going to the party!
Peter Double StartsWith Pete Double EndsWith eter Party!	Peter, Peter, Peter, Peter are going to the party!
Peter Remove StartsWith P Party!	Nobody is going to the party!

10. PARTY RESERVATION FILTER MODULE



You need to implement a filtering module to a party reservation software. First, the Party Reservation Filter Module (PRFM for short) has been **passed a list** with invitations. Next, the PRFM receives a **sequence of commands** that specify whether you need to add or remove a given filter.

Each PRFM command is in the given format:

`"{command;filter type;filter parameter}"`

You can receive the following PRFM commands:

- "Add filter"
- "Remove filter"
- "Print"

The possible PRFM filter types are:

- "Starts with"
- "Ends with"
- "Length"
- "Contains"

All PRFM filter parameters will be a string (or an integer only for the "Length" filter). Each command will be valid e.g. you won't be asked to remove a non-existent filter. The input will **end** with a "Print" command, after which you should print all the party-goers that are left after the filtration. See the examples below:

EXAMPLES

Input	Output
Peter Misha Slav Add filter;Starts with;P Add filter;Starts with;M Print	Slav
Peter Misha John Add filter;Starts with;P Add filter;Starts with;M Remove filter;Starts with;M Print	Misha John

11. TRIFUNCTION



Create a program that traverses a collection of names and returns the **first name**, whose sum of characters is **equal** to or **larger** than a given number **N**, which will be given on the first line. Use a function that **accepts another function** as one of its parameters.

Start by building a regular function to hold the basic logic of the program. Something along the lines of **Func<string, int, bool>**. Afterward, create your main function which should accept the first function as one of its parameters.

EXAMPLES

Input	Output
350 Rob Mary Paisley Spencer	Mary
666 Paul Thomas William	William