# Exercises: Functional Programming

Problems for exercises and homework for the ["C# Advanced" course @ Software University.](#)

You can check your solutions here: [https://judge.softuni.bg/Contests/1473/Functional-Programming-Exercises](https://judge.softuni.bg/Contests/1473/Functional-Programming-Exercises)

## Problem 1.  Action Point

Write 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 |
|---|---|
| Pesho Gosho Adasha | Pesho<br>Gosho<br>Adasha |

## Problem 2.  Knights of Honor

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

### Examples

| Input | Output |
|---|---|
| Pesho Gosho Adasha StanleyRoyce | Sir Pesho<br>Sir Gosho<br>Sir Adasha<br>Sir StanleyRoyce |

## Problem 3.  Custom Min Function

Write 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 |

## Problem 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<br>odd | 1 3 5 7 9 |

| | |
|---|---|
| 20 30<br>even | 20 22 24 26 28 30 |

# Problem 5.  Applied Arithmetics

Write 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

Use functions.

## Examples

| Input | Output |
|---|---|
| 1 2 3 4 5<br>add<br>add<br>print<br>end | 3 4 5 6 7 |
| 5 10<br>multiply<br>subtract<br>print<br>end | 9 19 |

# Problem 6.  Reverse and Exclude

Write 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<br>2 | 5 3 1 |
| 20 10 40 30 60 50<br>3 | 50 40 10 20 |

# Problem 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**.

SoftUni Foundation

## Examples

| Input | Output |
|---|---|
| 4<br>Kurnelia Qnaki Geo Muk Ivan | Geo<br>Muk<br>Ivan |
| 4<br>Karaman Asen Kiril Yordan | Asen |

# Problem 8.  Custom Comparator

Write a custom comparator that sorts all even numbers before all the odd ones in ascending order. Pass it to **Array.Sort()** function and print the result. Use functions.

## Examples

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

# Problem 9.  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<br>1 1 1 2 | 2 4 6 8 10 |
| 100<br>2 5 10 20 | 20 40 60 80 100 |

# Problem 10.  Predicate Party!

Ivancho'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 with 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 a given **criteria**. There are **three different criteria**:

- Everyone that has his **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 there are **no guests** going to the party print "Nobody is going to the party!". See the examples below:

---

## Examples

| Input | Output |
|---|---|
| Pesho Misho Stefan<br>Remove StartsWith P<br>Double Length 5<br>Party! | Misho, Misho, Stefan are going to the party! |
| Pesho<br>Double StartsWith Pesh<br>Double EndsWith esho<br>Party! | Pesho, Pesho, Pesho, Pesho are going to the party! |
| Pesho<br>Remove StartsWith P<br>Party! | Nobody is going to the party! |

# Problem 11.  Party Reservation Filter Module

You need to implement a filtering module to a party reservation software. First, to the Party Reservation Filter Module (PRFM for short) is **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 |
|---|---|
| Pesho Misho Slav<br>Add filter;Starts with;P<br>Add filter;Starts with;M<br>Print | Slav |
| Pesho Misho Jica<br>Add filter;Starts with;P<br>Add filter;Starts with;M | Misho Jica |

| | |
|---|---|
| Remove filter;Starts with;M<br>Print | |

## Problem 12. TriFunction

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

### Examples

| Input | Output |
|---|---|
| 800<br>Qvor Qnaki Petromir Sadam | Petromir |