# Exercises: Functional Programming

This document defines the exercises for "Java Advanced" course @ Software University.

Please submit your solutions (source code) of all below described problems in Judge.

## 1. Consumer Print

Write a program that **reads** a collection of **strings**, separated by one or more whitespaces, from the console and then prints them onto the console. Each string should be printed on a new line. Use a **Consumer<T>**.

### Examples

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

## 2. Knights of Honor

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

### Examples

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

## 3. Custom Min Function

Write a simple program that **reads** a **set of numbers** from the console and finds the **smallest** of the **numbers** using a simple **Function<Integer[], Integer>** .

### Examples

| Input | Output |
|---|---|
| 1 4 3 2 1 7 13 | 1 |

## 4. Applied Arithmetic

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 numbers in the list: **"add"** -> adds 1; **"multiply"** -> multiplies by 2; **"subtract"** -> subtracts 1; **"print"** -> prints all numbers on **a new line**. The input will end with an "**end**" command, after which you need to print the result.

## Examples

| Input | Output |
|---|---|
| 1 2 3 4<br>5<br>add<br>add<br>print<br>end | 3 4 5<br>6 7 |

| Input | Output |
|---|---|
| 5 10<br>multiply<br>subtract<br>print<br>end | 9 19 |

## 5. Reverse and Exclude

Write a program that **reverses** a collection and **removes** elements that are **divisible** by a given integer **n**.

## Examples

| Input | Output |
|---|---|
| 1 2 3 4 5 6<br>2 | 5 3 1 |
| 20 10 40 30 60 50<br>3 | 50 40<br>10 20 |

## 6. Predicate for names

Write a **predicate**. Its goal is to **check** a name for its length and to return **true** if the names length is **less or equal** the passed **integer**. You will be given a **string** array with some names. Print the names, passing the **condition** in the predicate.

## Examples

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

## 7. Find the smallest element

Write a program which is using a custom **function** (written by you :)) to find the **smallest** integer in a **sequence** of integers. The input could have more than one space. Your task is to **collect** the integers from the console, find the **smallest one** and print its **index** (if more than one such elements exist, print the index of the rightmost one).

## Hints

- Use a **Function<List<Integer>, Integer>** or something similar.

## Examples

| Input | Output |
|---|---|
| 1 2 3 0 4 5 6 | 3 |
| 123 10 11 3 | 3 |

## 8. Custom Comparator

Write a custom **comparator** that **sorts** all even numbers before all **odd** ones in **ascending order**. Pass it to an **`Arrays.sort()`** function and print the result.

### Examples

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

## 9. List of Predicates

Find all **numbers** in the range **1..N** that are **divisible** by the numbers of a given sequence. Use **predicates**.

### 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 |

## 10. Predicate Party!

The Wire'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 get 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 **given criteria**. There are three different options:

- 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**.

If nobody is going, 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! |

## 11. * The Party Reservation Filter Module

You are a young and talented **developer**. The first task you need to do is to implement a **filtering module** to a party reservation software. First, The Party Reservation Filter Module (**TPRF** Module for short) is passed a **list** with invitations. Next the **TPRF** receives a sequence of **commands** that specify if you need to add or remove a given filter.

**TPRF** Commands are in the given format **{command;filter type;filter parameter}**

You can receive the following **TPRF** commands: **"Add filter"**, **"Remove filter"** or **"Print"**. The possible **TPRF** filter types are: **"Starts with", "Ends with", "Length"** and **"Contains".** All **TPRF** filter parameters will be a string (or an integer for the length filter).

The input will end with a **"Print"** command. 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<br>Remove filter;Starts with;M<br>Print | Misho Jica |

## 12. *Inferno III

Your game studio's next triple A big-budget-killer-app is the Hack and Slash Action RPG Inferno III. The main purpose of the game is well, to hack and slash things. But the secondary purpose is to craft items and recently the main fan base has started complaining that once you craft an item you can't change it. So your next job is to implement a feature for one time reforging an item.

On the first line you are given the gems already inserted in the form of numbers, representing their power. On the next lines, until you receive the "Forge" command, you can receive commands in the following format **{command;filter type;filter parameter}**:

Commands can be: "Exclude", "Reverse" or "Forge". The possible filter types are: "Sum Left", "Sum Right" and "Sum Left Right". All filter parameters will be an integer.

"Exclude" marks a gem for exclusion from the set if it meets a given condition. "Reverse" deletes a previous exclusion.

"Sum Left" tests if a gems power summed with the gem standing to its left gives a certain value. "Sum Right" is the same but looks to a gems right peer. "Sum Left Right" sums the gems power with both its left and right neighbours.

Note that changes on to the item are made only after forging. This means that the gems are fixed at their positions and every function occurs on the original set, so every gems power is considered, no matter if it is marked or not.

To better understand the problem, see the examples below:

## Examples

| Input | Output | Comments |
|---|---|---|
| 1 2 3 4 5<br>Exclude;Sum Left;1<br>Exclude;Sum Left Right;9<br>Forge | 2 4 | 1. Marks for exclusion all gems for which the sum with neighbors to their left equals 1, e.g. 0 + **1** = 1<br><br>2. Marks for exclusion all gems for which the sum with neighbors to their left and their right equals 9, e.g.<br>2 + **3** + 4 = 9<br>4 + **5** + 0 = 9 |
| 1 2 3 4 5<br>Exclude;Sum Left;1<br>Reverse;Sum Left;1<br>Forge | 1 2 3 4 5 | 1. Marks for exclusion all gems for which the sum with their gem peers to the left equals 1, e.g. 0 + 1 = 1<br><br>2. Reverses the previous exclusion. |