Exercises: Generics

Problems for exercises and homework for the "CSharp Advanced" course @ Software University.

You can check your solutions here: https://judge.softuni.bg/Contests/1475/Generics-Exercises

1. Generic Box of String

Create a generic **`class Box`** that can be initialized with **any type** and **store the value**. Override the **`ToString()`** method and **print** the **type** and **stored value** in **format:**

```
"{class full name: value}"
```

On the first line, you will get **n** - the number of strings to read from the console. On the next **n** lines, you will get the **actual strings**. For each of them, create a box and call its **`ToString()`** method to **print** its data on the console.

Examples

Input	Output
2 life in a box box in a life	System.String: life in a box System.String: box in a life

2. Generic Box of Integer

Use the description of the previous problem but now, test your generic box with Integers.

Examples

Input	Output
3 7 123 42	System.Int32: 7 System.Int32: 123 System.Int32: 42

3. Generic Swap Method Strings

Create a **generic method** that receives **a list**, containing **any type** of **data** and **swaps the elements** at **two given indexes**. As in the previous problems, read **n** number of boxes of type **string** and **add** them to the list. On the next line, however, you will receive a **swap command** consisting of **two indexes**. Use the **method** you've created to swap the elements that correspond to the given indexes and **print each element in the list**.

Examples

Input	Output
3 Pesho Gosho Swap me with Pesho 0 2	System.String: Swap me with Pesho System.String: Gosho System.String: Pesho

4. Generic Swap Method Integers

Use the description of the previous problem, but now, **test** your list of generic boxes with **`integers`**.

SoftUni Foundation
© Software University Foundation. This work is licensed under the CC-BY-NC-SA license.
Follow us:

Examples

Input	Output
3 7 123 42 0 2	System.Int32: 42 System.Int32: 123 System.Int32: 7

5. Generic Count Method Strings

Create a **method** that receives as an argument a **list of any type, that can be compared** and an **element of the given type**. The method should **return the count of elements that are greater than the value of the given element**. **Modify your Box `class`** to support **comparison by value** of the stored data. On the **first** line, you will receive **n** - the number of **elements** to **add to the list**. On the next **n** lines, you will receive the **actual elements**. On the **last** line, you will get the **value** of the **element** to which you need to **compare every element** in the list.

Examples

Input	Output
3 aa aaa bb aa	2

6. Generic Count Method Doubles

Use the description of the previous problem, but now, **test** your **list** of generic boxes with **`doubles`**.

Examples

Input	Output
3 7.13 123.22 42.78 7.55	2

7. Tuple

A **`Tuple`** is a class in C#, in which you can store a few objects. First, we are going to focus on the type of **`Tuple`**, which contains two objects. The first one is **`"item1"`** and the second one is **`"item2"`**. It is kind of like a **`KeyValuePair`**, except – it **simply has items,** which are **neither key nor value**. Your task is to create a class **`"Tuple"`**, which holds two objects. The first one, will be **`"item1"`** and the second one – **`"item2"`**. The tricky part here is to make the class **hold generics**. This means, that when you create a new object of class – **`"Tuple"`**, there should be a way to explicitly, specify both the items' **type separately**.

Input

The input consists of **three** lines:

- The **first** one is holding a **person's name** and **an address**. They are separated by space(s). Your task is to **collect** them in the **tuple** and **print** them on the **console**. Format of the input:
  `{first name} {last name} {address}`
- The second line holds a **name** of a person and the **amount of beer** (int) he can drink. Format:

SoftUni Foundation
© Software University Foundation. This work is licensed under the CC-BY-NC-SA license.
Follow us:

{name} {liters of beer}

- The last line will hold an **integer** and a **double**. Format:
  {integer} {double}

Output

- Print the tuples' items in format: **{item1} -> {item2}**

Constraints

- Use the good practices we have learned. Create the class and make it have getters and setters for its class variables. The input will be **valid**, no need to check it explicitly!

Examples

Input	Output
Adam Smith California Mark 2 23 21.23212321	Adam Smith -> California Mark -> 2 23 -> 21.23212321

8. Threeuple

Create a Class **Threeuple**. Its name is telling us, that it will hold no longer, just a pair of objects. The task is simple, our **Threeuple** should **hold three objects**. Make it have getters and setters. You can even extend the previous class

Input

The input consists of three lines:

- The first one is holding a name, an address and a town. Format of the input:
  {first name} {last name} {address} {town}
- The second line is holding a **name**, **beer liters**, and a **boolean** variable with value - **drunk** or **not**. Format:
  {name} {liters of beer} {drunk or not}
- The last line will hold a **name**, a **bank balance** (**double**) and a **bank name**. Format:
  {name} {account balance} {bank name}

Output

- Print the Threeuples' objects in format:
  "{firstElement} -> {secondElement} -> {thirdElement}"

Examples

Input	Output
Adam Smith Wallstreet New York Mark 18 drunk Karren 0.10 USBank	Adam Smith -> Wallstreet -> New York Mark -> 18 -> True Karren -> 0.1 -> USBank
Ivan Ivanov TheHills Plovdiv Mitko 18 not George 0.10 NGB	Ivan Ivanov -> TheHills -> Plovdiv Mitko -> 18 -> False George -> 0.1 -> NGB

Note

You may extend your previous solution.

9. Custom Linked List

Now you have the needed knowledge to extend the custom linked list you have created during the previous workshop and your task is to make it **generic**. Upload your solution without the bin and obj folders in Judge.

SoftUni Foundation
© Software University Foundation. This work is licensed under the CC-BY-NC-SA license.
Follow us: