EXercises: Generics

You can check your solutions here: https://judge.softuni.bg/Contests/3181/Generics.

1. Generic Swap Method Integers

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

Examples

Input	Output

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

2. Tuple

A **Tuple** is a class in C#, in which you can store a few objects. First, we are going to focus on the **Tuple's type**, 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 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:
  `{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

3. 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.