Generics- Exercises

1. Jar of T 1
2. Generic Array Creator 1
3. Generic Scale 1
4. Generic Box 2
5. Generic Box of Integer 2
6. Generic Swap Method Strings 3
7. Generic Swap Method Integers 3
8. Generic Count Method Strings 4
9. Generic Count Method Doubles 4
10. Custom List 4

1. Jar of T

Create a class **Jar<>** that can store **anything**.

It should have two public methods:

- **void Add(element)**
- **element Remove()**

Adding should add on **top** of its contents. Remove should get the **topmost** element.

Use the syntax **Jar<T>** to create a generic class.

2. Generic Array Creator

Create a class **ArrayCreator** with a method and a single overload to it:

- **static T[] Create(int length, T item)**
- **static T[] Create(Class<T> class, int length, T item)**

The method should return an array with the given length, and every element should be set to the given default item.

3. Generic Scale

Create a class **Scale<T> where T : IComparable<T>** that holds two elements - **left** and **right**. The scale should receive the elements through its single constructor:

- **Scale(T left, T right)**

The scale should have a single method:

- **T GetHeavier()**

The **greater** of the two elements is heavier. The method should return **null** if elements are **equal**. Use **IComparable<T>** so that every T is extending Comparable, which gives us a compareTo() method.

4. Generic Box

Create a **generic class Box** that can store any type. **Override** the **ToString()** method to print the type and the value of the stored data in the format "**{class full name}: {value}**".

Use the class that you've created and test it with some class. 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.

Input	Output
2 **chicken in a box** **cat in a box**	System.String: chicken in a box System.String: cat in a box
1 **C#**	System.String: C#

5. Generic Box of Integer

Test your generic box with **Integers**.

Input	Output
3 **7** **123** **42**	System.Int32: 7 System.Int32: 123 System.Int32: 42
5 **12** **13**	System.Int32: 12 System.Int32: 13 System.Int32: 14

14 **15** **16**	System.Int32: 15 System.Int32: 16

6. 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.

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 corresponding to the given indexes and **print each** element in the list.

Input	Output
3 **Peter** **George** **Swap me with Peter** **0 2**	System.String: Swap me with Peter System.String: George System.String: Peter

7. Generic Swap Method Integers

Test your list of generic boxes with **Integers**.

Input	Output
3 **7** **123** **42** **0 2**	System.Int32: 42 System.Int32: 123 System.Int32: 7
5 **12** **13** **14** **15**	System.Int32: 12 System.Int32: 13 System.Int32: 14 System.Int32: 16 System.Int32: 15

16 **3 4**	

8. 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 **comparing by the value** of the data stored.

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 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	Input	Output
3 **aa** **aaa** **bb** **aa**	2	**3** **a** **b** **c** **d**	0

9. Generic Count Method Doubles

Test your list of generic boxes with **Doubles**.

Examples

Input	Output	Input	Output
3 **7.13** **123.22** **42.78** **7.55**	2	1 1231542.123 1	1

10. Custom List

Create a generic data structure that can store **any type** that can be **compared.** Implement functions:

- **void Add(T element)**
- **T Remove(int index)**
- **bool Contains(T element)**
- **void Swap(int index, int index)**
- **int CountGreaterThan(T element)**
- **T GetMax()**
- **T GetMin()**

Create a command interpreter that reads commands and modifies the custom list that you have created. Implement the commands:

- **Add {element}** - Adds the given element to the end of the list.
- **Remove {index}** - Removes the element at the given index.
- **Contains {element}** - Prints if the list contains the given element (**true** or **false**).
- **Swap {index1} {index2}** - Swaps the elements at the given indexes.
- **Greater {element}** - Counts the elements that are greater than the given element and prints their count
- **Max** - Prints the maximum element in the list.
- **Min** - Prints the minimum element in the list.
- **Print** - Prints all elements in the list, each on a separate line.
- **end** - stops the reading of commands.

Input	**Output**	**Input**	**Output**
Add aa	cc	Add Peter	Bobby Peter
Add bb	aa	Add Bobby	
Add cc	2	Swap 0 0	
Max	true	Swap 1 1	
Min	cc	Swap 0 1	
Greater aa	bb	Swap 1 0	
Swap 0 2	aa	Swap 0 1	
Contains aa		Print	
Print		end	

end			