# More Exercises: Data Types and Variables

## Data Type Finder

You will receive an input until you receive "END". Find what **data type** is the input. Possible data types are:

* Integer
* Floating point
* Characters
* Boolean
* Strings

Print the result in the following format: "**{input}** is **{data type}** type"

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  2.5  true  END | 5 is integer type  2.5 is floating point type  true is boolean type |
| a  asd  -5 | a is character type  asd is string type  -5 is integer type |

## From Left to The Right

You will receive number which represent how many lines we will get as an input. On the next N lines, you will receive a string with 2 numbers separated by single space. You need to compare them. If the left number is greater than the right number, you need to print the sum of all digits in the left number, otherwise print the sum of all digits in the right number.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  1000 2000  2000 1000 | 2  1 |
| 4  123456 2147483647  5000000 -500000  97766554 97766554  9999999999 8888888888 | 46  5  49  90 |

## Exchange Integers

Read two integer numbers and after that **exchange their values** by using some programming logic. Print the variable values before and after the exchange, as shown below:

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  10 | Before:  a = 5  b = 10  After:  a = 10  b = 5 |

**Hints**

You may use a **temporary variable** to remember the old value of **a**, then assign the value of **b** to **a**, then assign the value of the temporary variable to **b**.

## Floating Equality

Write a program that safely compares floating-point numbers (double) with precision eps = 0.000001. Note that we cannot directly compare two floating-point numbers a and b by a==b because of the nature of the floating-point arithmetic. Therefore, we assume two numbers are equal if they are more closely to each other than some fixed constant eps.

You will receive two lines, each containing a floating-point number. Your task is to compare the values of the two numbers.

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **Number a** | **Number b** | **Equal (with precision eps=0.000001)** | **Explanation** |
| 5.3 | 6.01 | False | The difference of 0.71 is too big (> eps) |
| 5.00000001 | 5.00000003 | True | The difference 0.00000002 < eps |
| 5.00000005 | 5.00000001 | True | The difference 0.00000004 < eps |
| -0.0000007 | 0.00000007 | True | The difference 0.00000077 < eps |
| -4.999999 | -4.999998 | False | Border case. The difference 0.0000001== eps. We consider the numbers are different. |
| 4.999999 | 4.999998 | False | Border case. The difference 0.0000001 == eps. We consider the numbers are different. |

## Refactoring: Prime Checker

You are given a program that checks if numbers in a given range [2...N] are prime. For each number is printed "{number} -> {true or false}". The code however, is not very well written. Your job is to modify it in a way that is easy to read and understand.

### Code

|  |
| --- |
| **Sample Code** |
| int \_\_\_Do\_\_\_ = int.Parse(Console.ReadLine());  for (int takoa = 2; takoa <= \_\_\_Do\_\_\_; takoa++)  {  bool takovalie = true;  for (int cepitel = 2; cepitel < takoa; cepitel++)  {  if (takoa % cepitel == 0)  {  takovalie = false;  break;  }  }  Console.WriteLine($"{takoa} -> {takovalie.ToString().ToLower()}");  } |

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5 | 2 -> true  3 -> true  4 -> false  5 -> true |