More Exercise: Data Types and Variables

Problems for exercises and homework for the "Technology Fundamentals" course @ SoftUni.

You can check your solutions in Judge.

1. 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 END	a is character type asd is string type -5 is integer type

From Left to the Right

You will receive a number which represents how many lines we will get as an input. On the next N lines, you will receive a string with 2 numbers separated by a 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 2
4 123456 2147483647 5000000 -500000 97766554 97766554 9999999999 8888888888	46 5 49 90

2. 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.

3. 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

```
Scanner chetec = new Scanner(System.in);

int ___Do___ = Integer.parseInt(chetec.nextLine());
for (int takoa = 2; takoa <= ___Do___; takoa++) {
    boolean takovalie = true;
    for (int cepitel = 2; cepitel < takoa; cepitel++) {
        if (takoa % cepitel == 0) {
            takovalie = false;
            break;
        }
    }
    System.out.printf("%d -> %b%n", takoa, takovalie);
}
```

Examples

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

4. Decrypting Messages

You will receive a **key** (**integer**) and **n** characters afterward. Add the key to each of the characters and append them to **message**. At the end print the message, which you decrypted.

Input

- On the **first line**, you will receive the **key**
- On the **second line**, you will receive **n** – the number of **lines**, which will **follow**
- On the next **n lines** – you will receive **lower** and **uppercase** characters from the **Latin** alphabet

Output

Print the **decrypted message**.

Constraints

- The **key** will be in the interval **[0…20]**
- **n** will be in the interval **[1…20]**
- The **characters** will always be **upper** or **lower**-case letters from the **English alphabet**
- You will receive **one letter** per **line**

Examples

Input	Output
3 7 P l c q R k f	SoftUni

Input	Output
1 7 C d b q x o s	Decrypt

5. Balanced Brackets

You will receive **n** lines. On **those lines**, you will receive **one** of the following:

- Opening bracket – "**(**",
- Closing bracket – "**)**" or
- **Random string**

Your task is to find out if the **brackets** are **balanced**. That means after every **closing** bracket should follow an **opening** one. Nested parentheses are **not valid**, and if **two consecutive opening brackets** exist, the expression should be marked as **unbalanced**.

Input

- On the **first line**, you will receive **n** – the number of lines, which will follow
- On the next **n** lines, you will receive "**(**", "**)**" or **another** string

Output

You have to print "**BALANCED**", if the parentheses are balanced and "**UNBALANCED**" otherwise.

Constraints

- **n** will be in the interval **[1…20]**
- The length of the stings will be between **[1…100]** characters

Examples

Input	Output
8 (5 + 10) * 2 + (5) -12	BALANCED

Input	Output
6 12 *) 10 + 2 - (5 + 10)	UNBALANCED