Lab: Data Types and Variables

Problems for in-class lab for the "C# Fundamentals" course @ SoftUni
You can check your solutions in Judge

I. Integer and Real Numbers

1. Convert Meters to Kilometers

You will be given an integer that will be a distance in meters. Create a program that converts meters to kilometers formatted to the second decimal point.

Examples

Input	Output
1852	1.85
798	0.80

2. Pounds to Dollars

Create a program that converts British pounds to US dollars formatted to the 3rd decimal point.

1 British Pound = 1.31 Dollars

Examples

Input	Output
80	104.800
39	51.090

3. Exact Sum of Real Numbers

Create a program to enter **n** numbers and calculate and print their **exact sum** (without rounding).

Examples

Input	Output
3 10000000000000000005 10	100000000000000015
2 0.00000000003 333333333333.3	333333333333.30000000003

Hints

Use **`Decimal`** to not lose precision.

II. Data Types and Type Conversion

4. Centuries to Minutes

Create a program to enter an integer number of **centuries** and convert it to **years**, **days**, **hours** and **minutes**.

Examples

Input	Output
1	1 centuries = 100 years = 36524 days = 876576 hours = 52594560 minutes
5	5 centuries = 500 years = 182621 days = 4382904 hours = 262974240 minutes

Hints

- Use appropriate data types to fit the result after each data conversion.
- Assume that a year has 365.2422 days on average (the Tropical year).

5. Special Numbers

A **number** is **special** when its **sum of digits is 5, 7 or 11**.

Write a program to read an integer **n** and for all numbers in the range **1…n** to print the number and if it is special or not (**True** / **False**).

Examples

Input	Output
7	1 -> False 2 -> False 3 -> False 4 -> False 5 -> True 6 -> False 7 -> True
15	1 -> False 2 -> False 3 -> False 4 -> False 5 -> True 6 -> False 7 -> True 8 -> False 9 -> False 10 -> False 11 -> False 12 -> False 13 -> False 14 -> True 15 -> False

Hints

To calculate the sum of digits of given number **num**, you might repeat the following: sum the last digit (**num % 10**) and remove it (**num = num / 10**) until **num** reaches **0**.

6. Reversed Chars

Create a program that takes 3 lines of characters and prints them in reversed order with a space between them.

Examples

Input	Output
A B C	C B A
1 L &	& L 1

7. Concat Names

Read two names and a delimiter. Print the names joined by the delimiter.

Examples

Input	Output
John Smith ->	John->Smith
Jan White <->	Jan<->White
Linda Terry =>	Linda=>Terry

8. Town Info

You will be given 3 lines of input. On the first line, you will be given the name of the town, on the second – the population and on the third the area. Use the correct data types and print the result in the following format:

"Town {town name} has population of {population} and area {area} square km".

Examples

Input	Output
Sofia 1286383 492	Town Sofia has population of 1286383 and area 492 square km.
Kaliningrad 437456 223	Town Kaliningrad has population of 437456 and area 223 square km.

9. Chars to String

Create a program that reads 3 lines of input. On each line, you get a single character. Combine all the characters into one string and print it on the console.

Examples

Input	Output
a b c	abc
% 2 o	%2o
1 5 p	15p

10. Lower or Upper

Create a program that prints whether a given character is upper-case or lower case.

Examples

Input	Output
L	upper-case
f	lower-case

III. Variables

11. Refactor Volume of Pyramid

You are given a **working code** that finds the **volume of a pyramid**. However, you should consider that the variables exceed their optimum span and have improper naming. Also, search for variables that **have multiple purposes**.

Hints

- **Reduce the span** of the variables by declaring them at the moment they receive a value, not before.
- Rename your variables to **represent their** real **purpose** (example: "`dul`" should become `length`, etc.).
- Search for variables that have multiple purposes. If you find any, **introduce a new variable**.

Sample Code

```
double dul, sh, V = 0;
Console.WriteLine("Length: ");
dul = double.Parse(Console.ReadLine());
Console.WriteLine("Width: ");
sh = double.Parse(Console.ReadLine());
Console.WriteLine("Heigth: ");
V = double.Parse(Console.ReadLine());
V = (dul + sh + V) / 3;
Console.WriteLine($"Pyramid Volume: {V:f2}");
```

12. Refactor Special Numbers

You are given a **working code** that is a solution to **Problem 5**. **Special Numbers**. However, the variables are **improperly named, declared before** they are needed and some of them are used for multiple purposes. Without using your previous solution, **modify the code,** so that it is **easy to read and understand**.

Sample Code

```
int kolkko = int.Parse(Console.ReadLine());
int obshto = 0;
int takova = 0;
bool toe = false;
for (int ch = 1; ch <= kolkko; ch++)
{
    takova = ch;
    while (ch > 0)
    {
        obshto += ch % 10;
        ch = ch / 10;
    }
    toe = (obshto == 5) || (obshto == 7) || (obshto == 11);
    Console.WriteLine("{0} -> {1}", takova, toe);
    obshto = 0;
    ch = takova;
}
```

Hints

- Reduce the span of the variables by declaring them at the moment they receive a value, not before.
- Rename your variables to represent their real purpose (example: "toe" should become **`isSpecialNum`**, etc.).
- Search for variables that have multiple purposes. If you find any, introduce a new variable.