# Exercises: Data Types and Variables

Problems for exercises and homework for the ["Technology Fundamentals" course @ SoftUni.](https://softuni.bg/courses/technology-fundamentals)

You can check your solutions here: <https://judge.softuni.bg/Contests/1228>

## Sum Digits

You will be given a single **integer**. Your task is to find the sum of its digits.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 245678 | 32 |
| 97561 | 28 |
| 543 | 12 |

## Chars to String

Write 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 |

## 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. |

## Convert Meters to Kilometres

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

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1852 | 1.85 |
| 798 | 0.80 |

## Pounds to Dollars

Write a program that converts British pounds to US dollars formatted to 3th decimal point.

1 British Pound = 1.31 Dollars

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 80 | 104.800 |
| 39 | 51.090 |

## Reversed Chars

Write 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 |

## Lower or Upper

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

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| L | upper-case |
| f | lower-case |

## \*Spice Must Flow

*Spice is Love, Spice is Life. And most importantly, Spice must flow. It must be extracted from the scorching sands of Arrakis, under constant threat of giant sand worms. To make the work as efficient as possible, the Duke has tasked you with the creation of a management software.*

Write a program that calculates the **total amount** of spice that can be extracted from a source. The source has a **starting yield**, which indicates how much spice can be mined on the **first day**. After it has been mined for a day, the **yield drops** by 10, meaning on the second day it’ll produce 10 less spice than on the first, on the third day 10 less than on the second, and so on (see examples). A source is considered profitable only while its yield is **at least** 100 – when less than 100 spice is expected in a day, abandon the source.

The mining crew **consumes** 26 spice **every day** at the end of their shift and **an additional** 26 after the mine has been exhausted. Note that the workers cannot consume more spice than there is in storage.

When the operation is complete, print on the console on two separate lines how many days the mine has operated and the total amount of spice extracted.

### Input

You will receive a **number**, representing the **starting yield** of the source.

### Output

Print on the console on two separate lines how many **days** the mine has operated and the **total amount** of spice extracted.

### Constraints

* The starting yield will be a positive **integer** within range [0 … 2 147 483 647]

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Explanation** |
| 111 | 2  134 | **Day 1** we extract 111 spice and at the end of the shift, the workers consume 26, leaving 85. The yield drops by 10 to 101.  **Day 2** we extract 101 spice, the workers consume 26, leaving 75. The total is 160 and the yield has dropped to 91.  **Since** the expected yield is less than 100, we abandon the source. The workers take another 26, leaving 134. The mine has operated 2 days. |

## \*Poke Mon

A Poke Mon is a special type of pokemon which likes to Poke others. But at the end of the day, the Poke Mon wants to keeps statistics, about how many pokes it has managed to make.

The Poke Mon pokes his target, and then proceeds to poke another target. The **distance** between his **targets** **reduces** his **poke power**.

You will be **given** the **poke power** the Poke Mon has, **N** – an **integer**.

Then you will be **given** the **distance** between the **poke targets**, **M** – an **integer**.

Then you will be **given** the **exhaustionFactor** **Y** – an **integer**.

Your task is to start **subtracting** **M** from **N** until **N** becomes **less than M**, i.e. the Poke Mon does not have enough power to reach the next target.   
**Every time** you **subtract M** from **N** that means you’ve reached a **target** and poked it successfully. **COUNT** how **many targets** you’ve poked – **you’ll need** that **count**.

The Poke Mon becomes gradually more exhausted. **IF** **N** **becomes equal** to **EXACTLY 50 %** of its **original value**, you must **divide** **N** by **Y**, if it is **POSSIBLE**. **This** **DIVISION** is between **integers**.

If a division is **not possible**, you should **NOT** do it. Instead, you should continue **subtracting**.

**After dividing**, you should **continue** subtracting from **N**, until it becomes **less** than **M**.

When **N** becomes **less** than **M**, you must take **what has remained** of **N** and the **count** of **targets** you’ve poked, and print them as output.

**NOTE**: When you are **calculating percentages**, you should be **PRECISE** at **maximum**.

**Example**: 505 is **NOT EXACTLY 50 %** from 1000, its **50.5 %**.

### Input

* The input consists of **3 lines**.
* On the **first line** you will receive **N** – an **integer**.
* On the **second line** you will receive **M** – an **integer**.
* On the **third line** you will receive **Y** – an **integer**.

### Output

* The output consists of **2 lines**.
* On the **first line** print **what has remained** of **N**, after **subtracting** from it.
* On the **second line** print the **count** of **targets**, you’ve managed to poke.

### Constrains

* The integer **N** will be in the **range** **[1, 2.000.000.000]**.
* The integer **M** will be in the **range** **[1, 1.000.000]**.
* The integer **Y** will be in the **range** **[0, 9]**.
* Allowed time / memory: **16 MB / 100ms**.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 5  2  3 | 1  2 | N = 5, M = 2, Y = 3.  We start **subtracting** **M** from **N**.  **N – M = 3**. **1** target poked.  **N – M = 1**. **2** targets poked.  **N < M**. We print **what has remained** of **N**, which is **1**.  We print the **count of targets**, which is 2. |
| 10  5  2 | 2  1 | N = 10, M = 5, Y = 2.  We start **subtracting** **M** from **N**.  **N** **–** **M** = **5**. (N is still not less than M, they are equal).  **N** became **EXACTLY** **50 %** of its **original value**.  **5** is **50 %** from **10**. So we divide **N** by **Y**.  **N / Y** = **5 / 2** = **2**. (**INTEGER DIVISION**). |

## \*Snowballs

Tony and Andi love playing in the snow and having snowball fights, but they always argue which makes the best snowballs. Because they are girls (which means they are completely illogical), they have decided to involve you in their fray, by making you write a program which calculates snowball data, and outputs the best snowball value.

You will receive N – an **integer**, the **number** of **snowballs** being made by Tony and Andi.  
**For each snowball** you will receive **3 input lines**:

* On the **first line** you will get the snowballSnow – an **integer**.
* On the **second line** you will get the snowballTime – an **integer**.
* On the **third line** you will get the snowballQuality – an **integer**.

**For each snowball** you must **calculate** its snowballValue by the following formula:

(snowballSnow / snowballTime) ^ snowballQuality

At the end you must print the **highest** calculated snowballValue.

### Input

* On the **first input line** you will receive **N** – the **number** of **snowballs**.
* On the **next N \* 3 input lines** you will be receiving **data** about **snowballs**.

### Output

* As output you must print the **highest** calculated snowballValue, by the formula, **specified above**.
* The output format is:   
  {snowballSnow} : {snowballTime} = {snowballValue} ({snowballQuality})

### Constraints

* The **number** of **snowballs** (N) will be an **integer** in **range [0, 100]**.
* The snowballSnow is an **integer** in **range [0, 1000]**.
* The snowballTime is an **integer** in **range [1, 500]**.
* The snowballQuality is an **integer** in **range [0, 100]**.
* Allowed working **time** / **memory**: **100ms** / **16MB**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  10  2  3  5  5  5 | 10 : 2 = 125 (3) |
| 3  10  5  7  16  4  2  20  2  2 | 10 : 5 = 128 (7) |