# Exercise: Data Types, Variables and Methods

Problems for exercises and homework for the [“Programming Fundamentals Extended” course @ SoftUni](https://softuni.bg/courses/programming-fundamentals).

## Anonymous Downsite

The Anonymous informal group of activists have hacked a few commercial websites and the CIA has hired you to write a software which calculates the losses. Based on the given data, use the appropiate data types.

You will receive **2 input lines** – each containing an **integer**.

* The **first** is **N** – the **number** of **websites** which are down.
* The **second** is the security key.

On the **next N lines** you will receive **data** about **websites** in the following format:  
{siteName} {siteVisits} {siteCommercialPricePerVisit}

You must **calculate** the site loss by the following formula: siteVisits \* siteCommercialPricePerVisit

When you **finish reading all data**, you must print the **affected sites’ names** – each on a **new line**.  
Then you must print the total money loss – **sum** of all site loss, on a **new line**.  
Finally you must print the security token, which is the security key, **POWERED** by the **COUNT** of **affected sites**.

### Input

* On the **first input line** you will get **N** – the **count** of **affected websites**.
* On the **second input line** you will the **security key**.
* On the **next N input lines** you will get **data** about the **websites**.

### Output

* As output you must print **all affected websites’ names** – **each** on a **new line**.
* **After** the **website names** you must print the **total loss** of **data**, printed to the **20th digit** after the **decimal point**. The format is “Total Loss: {totalLoss}”.
* Finally you must **print** the **security token**. The format is “Security Token: {securityToken}”.

### Constrains

* The integer **N** will be in **range** **[0, 100]**.
* The **security token** will be in **range** **[0, 10]**.
* The **website name** may contain any **ASCII character** except **whitespace**.
* The **site visits** will be an **integer** in **range [0, 231].**
* The **price per visit** will be a **floating point number** in **range [0, 100]** and will have **up** to **20 digits** after the decimal point.
* Allowed working **time/memory**: **100ms / 16MB**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  8  www.google.com 122300 94.23233  www.abv.bg 2333 11  www.kefche.com 12322 23.3222 | www.google.com  www.abv.bg  www.kefche.com  Total Loss: 11837653.10740000000000000000  Security Token: 512 |
| 1  1  www.facebook.com 100000 10.45 | www.facebook.com  Total Loss: 1045000.00000000000000000000  Security Token: 1 |

## Hornet Wings

The hornets are having a cardio contest. Your task is to calculate a contestant’s distance travelled, based upon the wing flaps he made. However some hornet contestants are faster and less durable, while others are slower but have more endurance.

You will be given **N** – an **integer** indicating the **wing flaps**, a contestant has chosen to do.

After that, you will receive **M** – a **floating-point number** indicating the **distance**, in **meters**, the hornet travels for **1000 wing flaps**.

Then you will receive **P** – an **integer** indicating the **endurance** of the contestant, or **how many wing flaps** he can make, before **he stops to take a break** and rest. A hornet **rests** for **5** **seconds**.

You can assume that a hornet makes **100** wing flaps **per** **second**.

Your task is to **calculate** how much **distance** will the hornet **travel**, after it **flaps** its **wings** **N times**, and how much **time** it **took him**, to travel it. The **distance** is measured in **meters** and the time – in **seconds**.

### Input

* On the first input line you will receive N – the wing flaps, the hornet has chosen to do.
* On the second input line you will receive M – the distance the hornet travels for 1000 wing flaps.
* On the third input line you will receive P – the endurance of the hornet.

### Output

* As output you must print the total distance the hornet contestant has travelled, and the amount of time it took him.
* The output must be in the format of two lines:
* On the first output line you must print the distance: “**{metersTraveled} m.**”
* On the second output line you must print the time: “**{secondsPassed} s.**
* The **distance** must be **printed** to the **second digit** after the **decimal point**.

### Constrains

* The integer **N** – the wing flaps, will be in **range [0; 1,000,000,000]**.
* The floating-point number **M** – the distance for 1000 wing flaps, will be in **range [0; 1,000,000]**.
* The integer **P** – the endurance, will be in range **[1; N]**.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 2000  5  200 | 10.00 m.  70 s. | The contestant has chosen to do **2000 wing flaps**.  He moves **5 meters** per **1000** wing flaps.  He rests every **200** **wing flaps** for **5** **seconds**.  The distance is **(2000 / 1000) \* 5 = 2 \* 5 = 10.00** meters.  The hornet flaps **100 times** for a **second**, so **2000 / 100 =** **20** seconds.  But it also rests for **5** seconds every **200** flaps.  **(2000 / 200) \* 5 = 10 \* 5** = **50**; **20** + **50** = **70** seconds. |
| 1000000  10  1500 | 10000.00 m.  13330 s. |

## 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**). |

## Resurrection

You ever heard of Phoenixes? Magical Fire Birds that are practically immortal – they reincarnate from an egg when they die. Naturally, it takes time for them to reincarnate. You will play the role of a scientist who calculates the time to reincarnate for each phoenix, based on its body parameters.

You will receive **N**, an **integer** – the **amount** of **phoenixes**.   
For each **phoenix**, you will **receive 3 input lines**:

* On the **first input line** you will receive an **integer** – the **total length** of the **body** of the phoenix.
* On the **second input line** you will receive a **floating-point number** – the **total width** of the **body** of the phoenix.
* On the **third input line** you will receive an **integer** – the **length** of **1 wing** of the phoenix.

For each phoenix, you must **print** the **years** it will take for it to **reincarnate**, which is **calculated** by the following formula:

The totalLength **powered** by 2, **multiplied** by the **sum of the** totalWidth and the totalWingLength (2 \* wingLength).

totalYears = {totalLength} ^ 2 \* ({totalWidth} + 2 \* {wingLength})

### Input

* On the **first input line** you will receive **N**, an **integer** – the **amount** of **phoenixes**.
* On the **next** **N \* 3 input lines** you will be receiving **data** for **each phoenix**.

### Output

* As output, you must print the **total years needed for reincarnation** for each phoenix.
* Print each phoenix’s years **when you’ve calculated** them.
* Print each phoenix’s years **on a new line**.

### Constrains

* The **amount** of **phoenixes** will be an **integer** in **range [0, 1000]**.
* The **total length** of the **body** of the **phoenix** will be an **integer** in **range [-231, 231]**.
* The **total width** of the **body** of the **phoenix** will be a **floating-point number** in **range [-231, 231]**.
* The **total width** of the **body** of the **phoenix** will have up to **20 digits** after the **decimal point**.
* The **total length** of the **wing** of the **phoenix** will be an **integer** in **range [-231, 231 – 1]**.
* The **total years** is a **product** of **integers** and **floating-point numbers**, thus it is a **floating-point number**.
* The **total years** should have the **same accuracy** as the **total width**.
* Allowed working time / memory: **100ms / 16MB**.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 2  100  50  30  150  25  10 | 1100000  1012500 | **2 phoenixes:**  **P1**:  Body length: 100  Body width: 50  Length of 1 wing: 30  Total years: 100 ^ 2 \* (50 + 2 \* 30) = 1100000  **P2**:  Body length: 150  Body width: 25  Length of 1 wing: 10  Total years: 150 ^ 2 \* (25 + 2 \* 10) = 1012500 |
| 2  100  50.243  31  154  23.132  11 | 1122430.000  1070350.512 | **2 phoenixes:**  **P1**:  Body length: 100  Body width: 50.243  Length of 1 wing: 31  Total years: 100 ^ 2 \* (50.243 + 2 \* 31) = 1122430.000  **P2**:  Body length: 154  Body width: 23.132  Length of 1 wing: 11  Total years: 154 ^ 2 \* (23.132 + 2 \* 11) = 1070350.512 |

## Wormtest

The worms are having a contest – a Wormtest. In the Wormtest, every worm is given wormpoints depending on several statistics. Guess who’s going to calculate those statistics... Yup, that’s you!

You will be given input data about a single wormtestant. You must calculate his wormpoints, depending on the given data.

On the **first** line of input you will get the worm’s **length** in **meters** (**m**), which will be an **integer**.

On the **second** line of input you will get the worm’s **width**, in **centimeters** (**cm**), which will be a **floating-point** **number**.

You should **convert** the **length** in **centimeters** (**multiply** it by **100**). Then you should **divide** the **length** and the **width** and find the **remainder** of the **division**.

If it is **zero** or **cannot be calculated**, you should **print** the **product** of **the length** and **the width** (**length** \* **width**), **rounded** to the **second digit** after the **decimal point**.

If the **remainder** is **NOT zero**, you should print what **percentage** is **the length** of **the width**.   
Print it **rounded** to the **second** **digit** after the **decimal point**.

**Example**: length = 1m ; width = 30cm. percentage = 333.33 %.

### Input

* On the **first** input line you will receive the worm’s **length** in **meters** (**m**).
* On the **second** input line you will receive the worm’s **width** in **centimeters** (**cm**).

### Output

* As output you must print the product of the **length** and the **width** or how much **percent** is the **length** of the **width**.
* **Both** output results should be **rounded** and **printed** to the **second digit** after the **decimal point**.

### Constrains

* The worm’s **length** will be a valid **integer** in **range [0, 1000]**.
* The worm’s **width** will be a **floating-point number** in **range [0, 1000.00]**.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 1000  200 | 20000000.00 | **length = 1000m. width = 200cm.**  **1000m \* 100 = 100000cm**.  **100000 % 200 = 0**. The **remainder** is **zero**. So we print the **product** of the two numbers, **rounded** to **the second digit** after the **decimal point**. |
| 40  600 | 666.67% | **40m \* 100 = 4000cm.**  **4000 % 600 = 400**. The **remainder** is **400**. So we print how much **percent** the **length** is of the **width**, **rounded** to **the second digit** after the **decimal point**.  **4000** is **666.67 percent** of **600**. |