

Exercise: Data Types and Variables

Please, submit your source code solutions for the described problems to the [Judge System](#).

Ask your questions here: <https://www.slido.com> by entering the course code **#fund-python**

1. Integer Operations

Write a program that reads **four integer numbers**. It should **add** the first to the second number, **integer divide** the sum by the third number, and **multiply** the result by the fourth number. Print the final result.

Examples

Input	Output	Input	Output
10 20 3 3	30	15 14 2 3	42

2. Chars to String

Write a function that receives **3 characters**. Concatenate 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

3. Elevator

Calculate how many courses will be needed to **elevate N persons** by using an elevator with a **capacity of P persons**. The input holds two lines: the **number of people N** and the **capacity P** of the elevator.

Examples

Input	Output	Comments
17 3	6	5 courses * 3 people + 1 course * 2 persons
4 5	1	All the people fit inside the elevator. Only one course is needed.

10 5	2	2 courses * 5 people
---------	---	----------------------

Hints

- You should **integer divide N by P**. This gives you the number of full courses (e.g., $17 / 3 = 5$).
- If **N** does not divide without a remainder, you will need one additional partially full course (e.g., $17 \% 3 = 2$).
- Another approach is to round up N / P to the nearest integer (ceiling), e.g., $17/3 = 5.67 \rightarrow$ rounds up to 6.
- For the round-up calculation, you might use `math.ceil()` function. Before you use it, you need to import the `math` library:

```
import math
```

```
courses = math.ceil(num_of_people/capacity)
```

4. Sum of Chars

Write a program, which **sums the ASCII codes of N characters** and prints the **sum** on the console. On the **first line**, you will receive **N** – the number of **lines**. On the following **N lines** – you will receive **a letter per line**. Print the **total sum** in the following format: **"The sum equals: {total_sum}"**.

Note: n will be in the interval [1...20].

Examples

Input	Output
5 A b C d E	The sum equals: 399

Input	Output
12 S o f t U n i R u l z z	The sum equals: 1263

5. Print Part of the ASCII Table

Write a program that **prints part of the ASCII table** characters on the console, separated by a single space. On the first line of input, you will receive **the char index you should start with**. On the **second line** - **the index of the last character** you should print.

Examples

Input	Output
60 65	< = > ? @ A
69 79	E F G H I J K L M N O

97 104	a b c d e f g h
40 55	() * + , - . / 0 1 2 3 4 5 6 7

6. Triples of Latin Letters

Write a program to read an integer N and print all **triples** of the first N small Latin letters, ordered alphabetically:

Examples

Input	Output	Input	Output
3	aaa aab aac aba abb abc aca acb acc baa bab bac bba bbb bbc bca bcb bcc caa cab cac cba cbb cbc cca ccb ccc	2	aaa aab aba abb baa bab bba bbb

Hints

- Perform 3 nested loops from 0 to N:

```
for i in range(0,num):
    for k in range(0,num):
        for j in range(0,num):
```

- For each iteration, you should generate new letters:

```
print(f"{chr( 97 + i )}{chr(97 + k )}{chr(97 + j )}")
```

7. Water Overflow

You have a **water tank** with a capacity of **255 liters**. On the **first line**, you will receive **n** – the number of **lines**, which will **follow**. On the following **n lines**, you will receive **liters of water** (integers), which you should **pour** into your **tank**. If the **capacity is not enough**, print **"Insufficient capacity!"** and **continue reading** the next line. On the last line, **print the liters** in the **tank**.

Examples

Input	Output	Input	Output
5 20 100 100 100 20	Insufficient capacity! 240	1 1000	Insufficient capacity! 0
7 10 20 30 10 5 10 20	105	4 250 10 20 40	Insufficient capacity! Insufficient capacity! Insufficient capacity! 250

8. * Party Profit

As a young adventurer, you travel with your group worldwide, seeking for gold and glory. But you need to split the profit among your companions.

You will receive a **group size**. After that, you receive the **days** of the adventure.

Every day, you earn **50 coins**, but you also spend **2 coins per companion** for food.

Every **3rd (third)** day, you organize a motivational party, spending **3 coins per companion** for drinking water.

Every **5th (fifth)** day, you slay a boss monster and **gain 20 coins per companion**. But if you have a motivational party the same day, you **spend an additional 2 coins per companion**.

Every **10th (tenth)** day at the start of the day, **2 (two)** of your companions **leave**, but every **15th (fifteenth)** day **5 (five)** **new** companions are joined at the beginning of the day.

You should calculate how many coins gets each companion at the end of the adventure.

Input / Constraints

The input will consist of **exactly 2 lines**:

- group size – integer in the range [1...100]
- days – integer in the range [1...100]

Output

Print the following message: "{companions_count} companions received {coins} coins each."

Note: You cannot split a coin, so you should round down the coins to an integer number.

Examples

Input	Output
3 5	3 companions received 90 coins each.
15 30	19 companions received 102 coins each.

9. *Snowballs

Tony and Andi love playing in the snow and having snowball fights, but they always argue about who makes the best snowballs. They have decided to involve you in their fray by writing a program that 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.

On the following lines, you will receive **3 inputs** for each **snowball**:

- The **weight** of the **snowball** (integer).
- The **time needed** for the snowball to get to its target (integer).
- The **quality** of the snowball (integer).

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

$$(\text{snowball_weight} // \text{snowball_time}) ** \text{snowball_quality}$$

In the end, you must print the **highest** calculated value of a snowball.

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 each **snowball**.

Output

- You need to print the **highest calculated snowball's value** in the format:
"{snowball_weight} : {snowball_time} = {snowball_value} ({snowball_quality})"

Constraints

- The **number** of **snowballs** (**N**) will be an **integer** in the **range** [0, 100].
- The weight is an **integer** in the **range** [0, 1000].
- The time is an **integer** in the **range** [1, 500].
- The quality is an **integer** in the **range** [0, 100].

Examples

Input	Output
-------	--------

2	10 : 2 = 125 (3)
10	
2	
3	
5	
5	
5	
3	10 : 5 = 128 (7)
10	
5	
7	
16	
4	
2	
20	
2	
2	

10. * Gladiator Expenses

As a gladiator, Peter needs to repair his broken equipment when he loses a fight. His equipment consists of a helmet, a sword, a shield, and armor.

You will receive Peter's **lost fights count**.

Every **second** lost game, his helmet is broken.

Every **third** lost game, his sword is broken.

When both **his sword and helmet are broken** in the same lost fight, his **shield also breaks**.

Every second time his shield breaks, his armor also needs to be repaired.

You will receive the price of each item in his equipment. Calculate his expenses for the year for renewing his equipment.

Input / Constraints

The input will consist of 5 **lines**:

- On the first line, you will receive the **lost fights count** – an integer in the range **[0, 1000]**.
- On the second line, you will receive the **helmet price** - a floating-point number in the range **[0, 1000]**.
- On the third line, you will receive the **sword price** - a floating-point number in the range **[0, 1000]**.
- On the fourth line, you will receive the **shield price** - a floating-point number in the range **[0, 1000]**.
- On the fifth line, you will receive the **armor price** - a floating-point number in the range **[0, 1000]**.

Output

- As output, you must print Peter's total expenses for new equipment: **"Gladiator expenses: {expenses} aureus"**

Examples

Input	Output	Comment
7 2 3 4 5	Gladiator expenses: 16.00 aureus	Trashed helmet -> 3 times Trashed sword -> 2 times Trashed shield -> 1 time Total: 6 + 6 + 4 = 16.00 aureus;
23 12.50 21.50 40 200	Gladiator expenses: 608.00 aureus	