# Exercises: Data Types and Variables

Problems for exercise and homework for the [Python Fundamentals Course @SoftUni](https://softuni.bg/trainings/3003/python-fundamentals-september-2020). Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/1722>

## Integer Operations

Read four integer numbers. Add first to the second, divide (integer) the sum by the third number and multiply the result by the fourth number. Print the result.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 10  20  3  3 | 30 |  | 15  14  2  3 | 42 |

## Chars to String

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

## Elevator

Calculate how many courses will be needed to **elevate n persons** by using an elevator with **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 persons fit inside the elevator.  Only one course is needed. |
| 10  5 | 2 | 2 courses \* 5 people |

### Hints

* You should **divide** n **by** p. This gives you the number of full courses (e.g. 17 / 3 = 5).
* If n does not divide p 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  rounds up to 6.
* For the round-up calculation you might use math.ceil() function. Before you can use it , you need to import math library:





## Sum of Chars

Write a program, which sums the ASCII codes of **n** characters and prints the **sum** on the console.

### Input

* On the **first** **line**, you will receive **n** – the number of **lines**, which will **follow**
* On the next **n lines** – you will receive letters from the **Latin** alphabet

### Output

Print the **total** **sum** in the following format:

The sum equals: {total\_sum}

### Constraints

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

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| **5**  A  b  C  d  E | The sum equals: 399 |  | **12**  S  o  f  t  U  n  i  R  u  l  z  z | The sum equals: 1263 |

## Print Part of the ASCII Table

Find online more information about [ASCII](http://www.ascii-code.com/) (American Standard Code for Information Interchange) and write a program that **prints part of the ASCII table** of characters on the console. On the first line of input you will receive **the char index you should start with** and 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 |

## 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** |
| 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 |

### Hints

Perform 3 nested loops from 0 to n.



For each iteration generate new letters



## Water Overflow

You have a **water** **tank** with capacity of **255 liters**. On the next **n** lines, you will receive **liters of water**, which you have to **pour** in 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**.

### Input

The **input** will be on two lines:

* On the **first** **line**, you will receive **n** – the number of **lines**, which will **follow**
* On the next **n lines** – you receive **quantities** of water, which you have to **pour** in the **tank**

### Output

Every time you do not have **enough** **capacity** in the tank to pour the given liters, **print**:

Insufficient capacity!

On the last line, **print** only the **liters** in the **tank**.

### Constraints

* **n** will be in the interval **[1…20]**
* **liters** will be in the interval **[1…1000]**

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| **5**  20  100  100  100  20 | Insufficient capacity!  240 | **1**  1000 | Insufficient capacity!  0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| **7**  10  20  30  10  5  10  20 | 105 | **4**  250  10  20  40 | Insufficient capacity!  Insufficient capacity!  Insufficient capacity!  250 |

## \* Party Profit

*As a young adventurer, you travel with your party around the world, seeking for gold and glory. But you need to split the profit among your companions.*

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

**Every day,** you are **earning 50 coins**, but you also spent **2 coins per companion** for food.

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

Every **5th (fifth)** day you slay a boss monster and you **gain 20 coins per companion**. But if you have a motivational party the same day, you **spent 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 have to calculate how much coins gets each companion at the end of the adventure.

### Input / Constraints

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

* party size – **integer in range [1…100]**
* days **– integer in range [1…100]**

### Output

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

You cannot split a coin, so take the integral part (round down the coins to integer number).

### Examples

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

## \*Snowballs

*Tony and Andi love playing in the snow and having snowball fights, but they always argue which makes the best snowballs. 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 snowball\_snow – an **integer**.
* On the **second line** you will get the snowball\_time – an **integer**.
* On the **third line** you will get the snowball\_quality – an **integer**.

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

(snowball\_snow / snowball\_time) ^ snowball\_quality

At the end you must print the **highest** calculated snowball\_value.

### 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 snowball\_value, by the formula, **specified above**.
* The output format is:   
  {snowball\_snow} : {snowball\_time} = {snowball\_value} ({snowball\_quality})

### Constraints

* The **number** of **snowballs** (N) will be an **integer** in **range [0, 100]**.
* The snowball\_snow is an **integer** in **range [0, 1000]**.
* The snowball\_time is an **integer** in **range [1, 500]**.
* The snowball\_quality is an **integer** in **range [0, 100]**.

### 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) |

## \* Gladiator Expenses

As a gladiator, Peter has to repair his broken equipment when he loses a fight. His equipment consists of helmet, sword, shield and armor. You will receive the 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 brakes**.

**Every** **second time**, when his shield brakes, 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** – integer in the range **[0, 1000]**.
* On the second line you will receive the **helmet price** - floating point number in range **[0, 1000]**.
* On the third line you will receive the **sword price** - floating point number in range **[0, 1000]**.
* On the fourth line you will receive the **shield price** - floating point number in range **[0, 1000]**.
* On the fifth line you will receive the **armor price** - floating point number in 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 |  |