# Exercise: Data Types and Variables

Problems for exercise and homework for the ["C# Fundamentals" course @ SoftUni](https://softuni.bg/trainings/3729/programming-fundamentals-with-csharp-may-2022)  
You can check your solutions in [Judge](https://judge.softuni.org/Contests/1205/Data-Types-and-Variables-Exercise)

## Integer Operations

Create a program that receives four integer numbers.

You should perform the following operations:

* **Add** first to the second.
* **Divide** (integer) the result of the first operation by the third number.
* **Multiply** the result of the second operation by the fourth number.

Print the result after the last operation.

### Constraints

* First number will be in the range [-2,147,483,648… 2,147,483,647]
* Second number will be in the range [-2,147,483,648… 2,147,483,647]
* Third number will be in the range [-2,147,483,648… 2,147,483,647]
* Fourth number will be in the range [-2,147,483,648… 2,147,483,647]

### Examples

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

## Sum Digits

Create a program that receives a single **integer**. Your task is to find the sum of its digits.

For example: **12345** -> **1+2+3+4+5 = 15**

### Examples

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

## Elevator

Calculate how many courses will be needed to **elevate n persons** by using an elevator of the **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 in 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.
* Sample code for the round-up calculation:



## Sum of Chars

Create 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: {totalSum}"

### 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 at 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 |
| 35  49 | # $ % & ' ( ) \* + , - . / 0 1 |

## Triples of Latin Letters

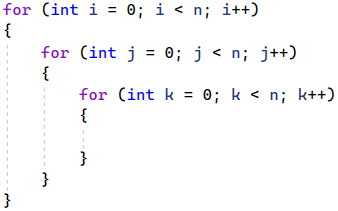
Create a program that receives an integer **n** and print all **triples** of the first **n small Latin letters**, ordered alphabetically:

### 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-1.



For each iteration generate new letters



## Water Overflow

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

### 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 will receive **quantities** of water, which you have to **pour** into 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 |

## Beer Kegs

Create a program, which calculates the volume of **n** beer kegs. You will receive in total **3 \* n** lines. **Every three lines** will hold **information** for a **single** keg. First up is the **model** of the keg, after that is the **radius** of the keg, and lastly is the **height** of the keg.

Calculate the volume using the following formula: π \* r^2 \* h.

In the end, print the **model** of the **biggest** keg.

### Input

You will receive **3 \* n** lines. Each group of lines will be on a new line:

* First – **model** – **string**.
* Second –**radius** – **floating-point** number
* Third – **height** – **integer** number

### Output

Print the **model** of the **biggest** keg.

### Constraints

* **n** will be in the interval **[1…10].**
* The **radius** will be a **floating-point number** in the interval **[1…3.402823E+38].**
* The **height** will be an **integer** in the interval **[1…2147483647].**

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| **3**  Keg 1  10  10  Keg 2  20  20  Keg 3  10  30 | Keg 2 |  | **2**  Smaller Keg  2.41  10  Bigger Keg  5.12  20 | Bigger Keg |