# Lab: 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/1227

# Integer and Real Numbers

## 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.

### 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 |

## Exact Sum of Real Numbers

Write program to enter n numbers and calculate and print their **exact sum** (without rounding).

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  1000000000000000000  5  10 | 1000000000000000015 |
| 2  0.00000000003  333333333333.3 | 333333333333.30000000003 |

### Hints

Use BigDecimal to not lose precision.

# Data Types and Type Conversion

## Elevator

Calculate how many courses will be needed to **elevate n persons** by using an elevator of **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:



## Centuries to Minutes

Write program to enter an integer number of **centuries** and convert it to **years**, **days**, **hours** and **minutes**.

### Examples

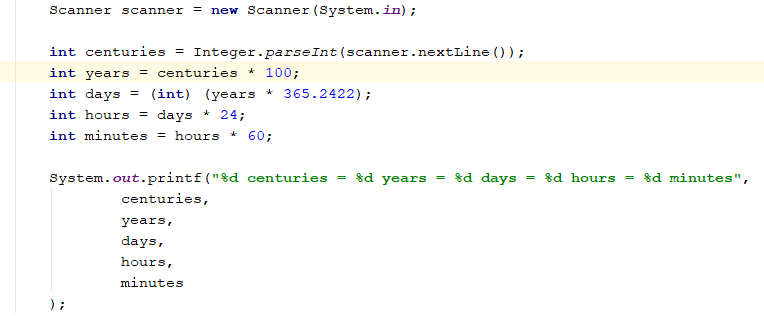
|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 | 1 centuries = 100 years = 36524 days = 876576 hours = 52594560 minutes |
| 5 | 5 centuries = 500 years = 182621 days = 4382904 hours = 262974240 minutes |

### Hints

* Use appropriate data type to fit the result after each data conversion.
* Assume that a year has 365.2422 days at average ([the Tropical year](https://en.wikipedia.org/wiki/Tropical_year)).

### Solution

You might help yourself with the code below:



## Special Numbers

A **number** is **special** when its **sum of digits is 5, 7 or 11**.

Write a program to read an integer n and for all numbers in the range **1…n** to print the number and if it is special or not (True / False).

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 15 | 1 -> False  2 -> False  3 -> False  4 -> False  5 -> True  6 -> False  7 -> True  8 -> False  9 -> False  10 -> False  11 -> False  12 -> False  13 -> False  14 -> True  15 -> False |

### Hints

To calculate the sum of digits of given number num, you might repeat the following: sum the last digit (num % 10) and remove it (sum = sum / 10) until num reaches 0.

## 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-1.



For each iteration generate new letters



Concat all characters in a string and print it. You can use String.format().



## Concat Names

Read two names and a delimiter. Print the names joined by the delimiter.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| John  Smith  -> | John->Smith |
| Jan  White  <-> | Jan<->White |
| Linda  Terry  => | Linda=>Terry |

# Variables

## Refactor Volume of Pyramid

You are given a **working code** that finds the **volume of a pyramid**. However, you should consider that the variables exceed their optimum span and have improper naming. Also, search for variables that **have multiple purpose**.

### Code

|  |
| --- |
| **Sample Code** |
| **double** dul, sh, V = 0; System.***out***.print(**"Length: "**); dul = Double.*parseDouble*(scanner.nextLine()); System.***out***.print(**"Width: "**); sh = Double.*parseDouble*(scanner.nextLine()); System.***out***.print(**"Heigth: "**); V = Double.*parseDouble*(scanner.nextLine()); V = (dul + sh + V) / 3; System.***out***.printf(**"Pyramid Volume: %.2f"**, V); |

### Hints

* **Reduce the span** of the variables by declaring them in the moment they receive a value, not before
* Rename your variables to **represent their** real **purpose** (example: "dul" should become length, etc.)
* Search for variables that have multiple purpose. If you find any, **introduce a new variable**.

## Refactor Special Numbers

You are given a **working code** that is a solution to **Problem 5. Special Numbers**. However, the variables are **improperly named, declared before** they are needed and some of them are used for multiple things. Without using your previous solution, **modify the code** so that it is **easy to read and understand**.

### Code

|  |
| --- |
| **Sample Code** |
| **int** kolkko = Integer.*parseInt*(scanner.nextLine()); **int** obshto = 0; **int** takova = 0; **boolean** toe = **false**; **for** (**int** ch = 1; ch <= kolkko; ch++) {  takova = ch;  **while** (ch > 0) {  obshto += ch % 10;  ch = ch / 10;  }  toe = (obshto == 5) || (obshto == 7) || (obshto == 11);  System.***out***.printf(**"%d -> %b%n"**, takova, toe);  obshto = 0;  ch = takova; } |

### Hints

* Reduce the span of the variables by declaring them in the moment they receive a value, not before
* Rename your variables to represent their real purpose (example: "dul" should become length, etc.)
* Search for variables that have multiple purpose. If you find any, introduce a new variable