**Exercises: Data Types and Methods**

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

You can check your solutions here: <https://judge.softuni.bg/Contests/171/Data-Types-and-Methods-Lab>.

* **Centuries to Minutes**

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

**Examples**

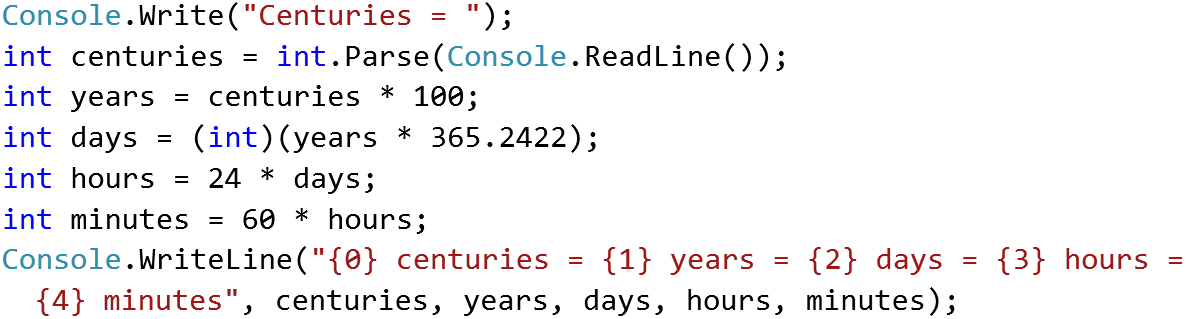
|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 | 1 centuries = 100 years = 36524 days = 876576 hours = 52594560 minutes |
| 5 | 5 centuries = 500 years = 182621 days = 4382904 hours = 262974240 minutes |

**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:



* **Circle Area (12 Digits Precision)**

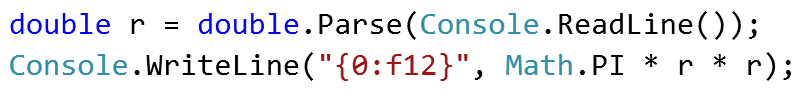
Write program to enter a radius **r** (real number) and **print the area of the circle** with exactly **12 digits** after the decimal point. Use data type of **enough precision** to hold the results.

**Examples**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 2.5 | 19.634954084936 |  | 1.2 | 4.523893421169 |

**Hints**

* You might use the data type **double**. It has precision of 15-16 digits.
* To print the output with exactly 12 digits after the decimal point, you might use the following code:



* **Exact Sum of Real Numbers**

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

**Examples**

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

**Hints**

* If you use types like **float** or **double**, the result will lose some of its precision. Also it might be printed in scientific notation.
* You might use the **decimal** data type which holds real numbers with high precision with less loss.
* Note that **decimal** numbers sometimes hold the unneeded zeroes after the decimal point, so **0m** is different than **0.0m** and **0.00000m**.
* **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:



* **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 number **num** print its corresponding Latin letter as follows:



* **Greeting**

Write a program that enters **first name**, **last name** and **age** and prints "***Hello, <first name> <last name>. You are <age> years old.***". Use interpolated strings.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| Svetlin  Nakov  25 | Hello, Svetlin Nakov. You are 25 years old. |

**Hints**

You might use the following code:



* **Day of Week**

Print the day name (in English) by day number in range [1...7] or print “**Error**” for invalid day number.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 | Monday |
| 2 | Tuesday |
| 3 | Wednesday |
| 4 | Thursday |
| 5 | Friday |
| 6 | Saturday |
| 7 | Sunday |
| -1 | Error |

**Hints**

Use the **switch-case** statement.

* **Animal Type**

Write a program to print **animal type by its name**:

* dog -> mammal
* crocodile, tortoise, snake -> reptile
* others -> unknown

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| dog | mammal |
| snake | reptile |
| cat | unknown |

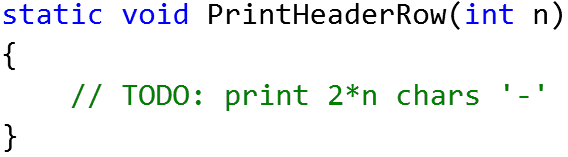
**Hints**

Use the **switch-case** statement.

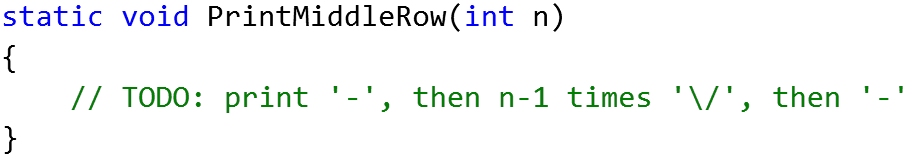
* **Filled Square**

Write a program to draw at the console a **filled square** of size **n** like in the examples below.

* First write a **method** to print the **header** / **footer** **rows**:



* Then write another **method** to print the **middle rows**:



* Print the entire filled square by **invoking the above two methods**.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3 | **------**  **-\/\/-**  **------** |
| 4 | **--------**  **-\/\/\/-**  **-\/\/\/-**  **--------** |