Lab: Data Types: Numeral Types and Type Conversion

Problems for exercises and homework for the "Programming Fundamentals" course @ SoftUni.

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

I. Integer and Real Numbers

1. Time Since Birthday

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

Examples

Input	Output
20	`20 years = 7300 days = 175200 hours = 10512000 minutes.`
14	`14 years = 5110 days = 122640 hours = 7358400 minutes.`

Hints

- Use appropriate data type to fit the result after each data conversion.
- Assume that every year has 365 days.

Solution

You might help yourself with the code below:

```
Console.Write("Years - ");
byte years = byte.Parse(Console.ReadLine());
int days = years * 365;
int hours = days * 24;
int minutes = hours * 60;

Console.WriteLine("{0} years = {1} days = {2} hours = {3} minutes.",
    years, days, hours, minutes);
```

2. Circle Perimeter (12 Digits Precision)

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

Examples

Input	Output
0.05	`0.314159265359`

Input	Output
1.2	`7.539822368616`

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:

```
double r = double.Parse(Console.ReadLine());
Console.WriteLine("{0:f12}", 2 * Math.PI * r);
```

3. Exact Product of Real Numbers

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

Examples

Input	Output
3 1000000000000000000 5 10	`50000000000000000000`

Input	Output
2 0.00000000003 333333333333.3	`9.999999999999`

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`**.

II. Type Conversion

4. Transport

Calculate how many courses will be needed to **transport n persons** by using 3 vehicles of **capacity 4, 8 and 12** respectively. The input holds one line: the **number of people n**. The vehicles **can** travel at the same time.

Examples

Input	Output	Comments

50	3	`2 course * 24 persons`
		`+ 1 course * 2 person`

24	1	`All the persons fit inside in one total course of the vehicles.`
		`Only one course is needed.`

150	7	`150 / (4 + 8 + 12) = 6.25 => 7 courses`
		`6 courses * 24 people (4 + 8 + 12) + 1 course * 6 people`

Hints

- You should **divide n by `the sum of all the cars' capacity`**. This gives you the number of full courses (e.g. 25 / 24 = 1.04).
- If **n** does not divide without a remainder, you will need one additional partially full course (e.g. 25 % 24 = 1).
- Another approach is to round up **n / `(4+8+12)`** to the nearest integer (ceiling), e.g. 25/24 = 1.04 → rounds up to 2.
- Sample code for the round-up calculation:

```
var capacity = 4 + 8 + 12;
var courses = (int) Math.Ceiling((double)n / capacity);
```