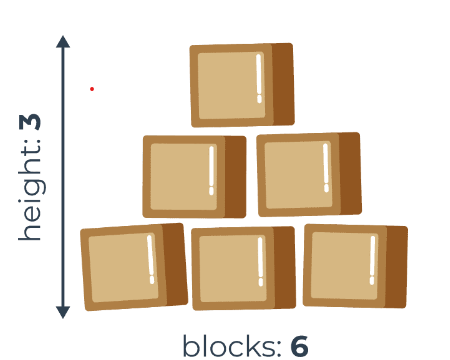
## Scenario

Listen to this story: a boy and his father, a computer programmer, are playing with wooden blocks. They are building a pyramid.

Their pyramid is a bit weird, as it is actually a pyramid-shaped wall – it's flat. The pyramid is stacked according to one simple principle: each lower layer contains one block more than the layer above.

The figure illustrates the rule used by the builders:



Your task is to write a program which reads the number of blocks the builders have, and outputs the height of the pyramid that can be built using these blocks.

Note: the height is measured by the number of **fully completed layers** – if the builders don't have a sufficient number of blocks and cannot complete the next layer, they finish their work immediately.

Test your code using the data we've provided.

## Test Data:

**Sample input:**

6

Expected output:

The height of the pyramid: 3

**Output**

**Sample input:**

20

Expected output:

The height of the pyramid: 3

**Output**

**Sample input:**

1000

Expected output:

The height of the pyramid: 44

**Output**

**Sample input:**

2

Expected output:

The height of the pyramid: 1

**Outp**

**Scenario**

[Spathiphyllum](https://upload.wikimedia.org/wikipedia/commons/b/bd/Spathiphyllum_cochlearispathum_RTBG.jpg), more commonly known as a peace lily or white sail plant, is one of the most popular indoor houseplants that filters out harmful toxins from the air. Some of the toxins that it neutralizes include benzene, formaldehyde, and ammonia.

Imagine that your computer program loves these plants. Whenever it receives an input in the form of the word **Spathiphyllum**, it involuntarily shouts to the console the following string: **"Spathiphyllum is the best plant ever!"**

Write a program that utilizes the concept of conditional execution, takes a string as input, and:

* prints the sentence **"Yes - Spathiphyllum is the best   
  plant ever!"** to the screen if the inputted string is **"Spathiphyllum"** (upper-case)
* prints **"No, I want a big Spathiphyllum!"** if the inputted string is **"spathiphyllum"** (lower-case)
* prints **"Spathiphyllum! Not [input]!"** otherwise. Note: **[input]** is the string taken as input.

Test your code using the data we've provided for you. And get yourself a Spathiphyllum, too!

**Test Data:**

**Sample input:**

spathiphyllum

**Expected output:**

No, I want a big Spathiphyllum!

**Output**

**Sample input:**

pelargonium

**Expected output:**

Spathiphyllum! Not pelargonium!

**Output**

**Sample input:**

Spathiphyllum

**Expected output:**

Yes - Spathiphyllum *is* the best plant ever!

**Output**

**Scenario**

Once upon a time there was a land – a land of milk and honey, inhabited by happy and prosperous people. The people paid taxes, of course – their happiness had limits. The most important tax, called the *Personal Income Tax* (*PIT* for short) had to be paid once a year, and was evaluated using the following rule:

* if the citizen's income was not higher than 85,528 thalers, the tax was equal to 18% of the income minus 556 thalers and 2 cents (this was what they called *tax relief*)
* if the income was higher than this amount, the tax was equal to 14,839 thalers and 2 cents, plus 32% of the surplus over 85,528 thalers.

Your task is to write a **tax calculator**.

* It should accept one floating-point value: the income.
* Next, it should print the calculated tax, rounded to full thalers. There's a function named **round()** which will do the rounding for you – you'll find it in the skeleton code in the editor.

Note: this happy country never returned any money to its citizens. If the calculated tax was less than zero, it would only mean no tax at all (the tax was equal to zero). Take this into consideration during your calculations.

Look at the code in the editor – it only reads one input value and outputs a result, so you need to complete it with some smart calculations.

Test your code using the data we've provided.

**Test Data**

**Sample input:**

10000

**Expected output:**

The tax *is*: 1244.0 thalers

**Output**

**Sample input:**

100000

**Expected output:**

The tax *is*: 19470.0 thalers

**Output**

**Sample input:**

1000

**Expected output:**

The tax *is*: 0.0 thalers

**Output**

**Sample input:**

-100

**Expected output:**

The tax *is*: 0.0 thalers

**Output**

**Scenario**

As you surely know, due to some astronomical reasons, years may be *leap* or *common*. The former are 366 days long, while the latter are 365 days long.

Since the introduction of the Gregorian calendar (in 1582), the following rule is used to determine the kind of year:

* if the year number isn't divisible by four, it's a *common year*;
* otherwise, if the year number isn't divisible by 100, it's a *leap year*;
* otherwise, if the year number isn't divisible by 400, it's a *common year*;
* otherwise, it's a *leap year*.

Look at the code in the editor – it only reads a year number, and needs to be completed with the instructions implementing the test we've just described.

The code should output one of two possible messages, which are **Leap year** or **Common year**, depending on the value entered.

It would be good to verify if the entered year falls into the Gregorian era, and output a warning otherwise: **Not within the Gregorian calendar period**. Tip: use the **!=** and **%** operators.

Test your code using the data we've provided.

**Test Data:**

**Sample input:**

2000

**Expected output:**

Leap year

**Output**

**Sample input:**

2015

**Expected output:**

Common year

**Output**

**Sample input:**

1999

**Expected output:**

Common year

**Output**

**Sample input:**

1996

**Expected output:**

Leap year

**Output**

**Sample input:**

1580

**Expected output:**

Not within the Gregorian calendar period

**Output**

**Scenario**

A junior magician has picked a secret number. He has hidden it in a variable named secret\_number. He wants everyone who runs his program to play the *Guess the secret number* game, and guess what number he has picked for them. Those who don't guess the number will be stuck in an endless loop forever! Unfortunately, he does not know how to complete the code.

Your task is to help the magician complete the code in the editor in such a way so that the code:

* will ask the user to enter an integer number;
* will use a while loop;
* will check whether the number entered by the user is the same as the number picked by the magician. If the number chosen by the user is different than the magician's secret number, the user should see the message "Ha ha! You're stuck in my loop!" and be prompted to enter a number again. If the number entered by the user matches the number picked by the magician, the number should be printed to the screen, and the magician should say the following words: "Well done, muggle! You are free now."

.