# Lab: Basic Syntax, Conditional Statements and Loops

You can check your solutions in [Judge.](https://judge.softuni.org/Contests/1190)

## Student Information

You will be given 3 lines of input – student name, age, and average grade. Your task is to print all the info about the student in the following format: "Name: {student name}, Age: {student age}, Grade: {student grade}".

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| John  15  5.40 | Name: John, Age: 15, Grade: 5.40 |
| Steve  16  2.50 | Name: Steve, Age: 16, Grade: 2.50 |
| Marry  12  6.00 | Name: Marry, Age: 12, Grade: 6.00 |

### Solution

First, we need a scanner, which we can use to read data from the console.



Read all the information – student name, age, and grade



Finally, we need to print the information in the specified format



## Passed

Write a program that takes as an input a **grade** and prints "**Passed!**" if the grade is **equal or more than 3.00**.

### Input

The **input** comes as a single floating-point number.

### Output

The **output** is either "**Passed!**" if the grade is **equal or more than 3.00**, otherwise, you should print nothing.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 5.32 | Passed! |  | 2.34 | *(no output)* |

## Passed or Failed

Modify the above program, so it will print "**Failed!**" if the grade is **lower than 3.00**.

### Input

The **input** comes as a single double number.

### Output

The **output** is either "**Passed!**" if the grade is **more than 2.99**, otherwise, you should print "**Failed!**".

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 5.32 | Passed! |  | 2.36 | Failed! |

### Hint

We need to take a **floating-point** number from the console. After that, print in the **else** statement the appropriate message.



## Back in 30 Minutes

Every time John tries to pay his bills, he sees on the cash desk the sign: **"I will be back in 30 minutes"**. One day John was sick of waiting and decided he needed a program that **prints the time** after **30** **minutes**. That way he won't have to wait at the desk and come at the appropriate time. He gave the assignment to you, so you have to do it.

### Input

The **input** will be on two lines. On the **first** **line**, you will receive the **hours,** and on the **second,** you will receive the **minutes**.

### Output

Print on the console the time after **30** minutes. The result should be in the format "**hh:mm"**. The **hours** have **one or two** **numbers,** and the **minutes** always have **two numbers (with leading zero)**.

### Constraints

* The **hours** will be between **0 and 23**.
* The **minutes** will be between **0 and 59**.

### Examples

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 1  46 | 2:16 |  | 0  01 | 0:31 | 23  59 | 0:29 |  | 11  08 | 11:38 |  | 11  32 | 12:02 |

### Hints

Add 30 minutes to the initial minutes that you receive from the console. If the minutes are more than 59 – increase the hours by 1 and decrease the minutes by 60. In the same way, check if the hours are more than 23. When you print, check for leading zero.

## Month Printer

Write a program that takes an **integer** from the console and prints the corresponding **month**. If the number **is more than 12** or **less than 1** print "**Error!**".

### Input

You will receive a **single** **integer** on a **single line**.

### Output

If the number is within the boundaries, print the corresponding month, otherwise, print "**Error!**".

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 2 | February |  | 13 | Error! |

### Hints



## Foreign Languages

Write a program that prints the language that a given country speaks. You can receive only the following combinations: English **is spoken** in England and USA; Spanish **is spoken** in Spain, Argentina, and Mexico; for the others**,** we should print "**unknown**".

### Input

You will receive a **single country name** on a **single line**.

### Output

**Print** the **language**, which the country **speaks**, or if it is **unknown** for your program, print **"**unknown**"**.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| USA | English |  | Germany | unknown |

### Hint

Think about how you can **merge** multiple cases to **avoid** writing more code than you need to.

## Theatre Promotions

A theatre **is having a ticket sale**, but they need a program **to** calculate the price of a single ticket. If the given age does not fit one of the categories**,** you should print "**Error!**". You can see the prices i**n** the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Day / Age** | **0 <= age <= 18** | **18 < age <= 64** | **64 < age <= 122** |
| **Weekday** | 12$ | 18$ | 12$ |
| **Weekend** | 15$ | 20$ | 15$ |
| **Holiday** | 5$ | 12$ | 10$ |

### Input

The input comes in **two lines**. On the **first** line, you will receive the **type of day**. On the **second** – is the **age** of the person.

### Output

Print the ticket price according to the table, or "**Error!**" if the age is not in the table.

### Constraints

* The age will be in the interval **[-1000…1000]**.
* The type of day will **always be** **valid**.

### Examples

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| Weekday  42 | 18$ |  | Holiday  -12 | Error! | Holiday  15 | 5$ |  | Weekend  122 | 15$ |

### Hints

We need to read **two** lines. **The first** one will be the **type of day**. We will convert it to **lower case** letters with the method "**toLowerCase()**". After that, we will read the person's **age** and declare a **variable** – **price**, which we will use to set the ticket price.



For every **type of day**, we will need to add **different cases** to check the person's **age** and **set the price**. Some **age groups** have **equal** **prices** for the **same type** of day. This means we can use **logical operators** to **merge some of the conditions**.



Think **where** and **how** you can use **logical operators** for the **other cases**.

We can check if the price **has a value** is different from the **initial** one. If it does, that means we got a **valid combination of day and age,** and the ticket price is saved in the price variable. If the price has a **value of 0**, then none of the cases got hit, therefore, we have to **print the error message**.



## Divisible by 3

Write a program that prints all the numbers from **1 to 100**, which are **divisible by 3**. You have to use a single for loop. The program should not receive input.

### Solution



## Sum of Odd Numbers

Write a program that prints the next **n** **odd numbers** (starting from 1) and on the **last row** prints the **sum of them**.

### Input

On the first line, you will receive a number – **n**. This number shows how many **odd numbers** you should print.

### Output

Print the next **n** odd numbers, starting from **1**, separated by **new lines**. On the last line, print the **sum** of these numbers.

### Constraints

* **n** will be in the interval **[1…100]**

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 5 | 1  3  5  7  9  Sum: 25 |  | 3 | 1  3  5  Sum: 9 |

### Hints



## Multiplication Table

You will receive an **integer** as input from the console. Print the **10 times table** for this integer. See the examples below for more information.

### Output

Print every row of the table in the following format:

{theInteger} X {times} = {product}

### Constraints

* The integer will be in the interval **[1…100]**

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 5 | 5 X 1 = 5  5 X 2 = 10  5 X 3 = 15  5 X 4 = 20  5 X 5 = 25  5 X 6 = 30  5 X 7 = 35  5 X 8 = 40  5 X 9 = 45  5 X 10 = 50 |  | 2 | 2 X 1 = 2  2 X 2 = 4  2 X 3 = 6  2 X 4 = 8  2 X 5 = 10  2 X 6 = 12  2 X 7 = 14  2 X 8 = 16  2 X 9 = 18  2 X 10 = 20 |

## Even Number

Take as an input an even number and **print its absolute value** with a message**:** "**The number is: {absoluteValue}**". If the number is odd, print "**Please write an even number.**" and continue reading numbers.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 1  3  6 | Please write an even number.  Please write an even number.  The number is: 6 |  | -6 | The number is: 6 |

## Refactor Sum of Odd Numbers

You are assigned to **find and fix the bugs** in an existing piece of code, using the **debugger**. You should trace the program execution to find the lines of code that produce incorrect or unexpected results.

You are given a program (existing source code) that prints the next **n** **odd numbers** (starting from 1) and on the **last row**, prints the **sum of them**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 5 | 1  3  5  7  9  Sum: 25 |  | 3 | 1  3  5  Sum: 9 |

|  |
| --- |
| SumOddNumbers.java |
| Scanner sc = **new** Scanner(System.***in***); **int** n = Integer.*parseInt*(sc.nextLine()); **int** sum = 1; **for** (**int** i = 0; i <= n; i++) {  System.***out***.print(2 \* i + 1);  sum += 2 \* i; } System.***out***.printf(**"Sum: %d%n"**, sum); |