Lab: Intro and Basic Syntax

Problems for in-class lab for the "C# Fundamentals" course @ SoftUni
You can check your solutions in Judge

1. Student Information

Create a program that receives 3 lines of input:

- student name
- age
- average grade

Your task is to print all of 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`

2. Passed

Create a program that receives a single number as an input representing a grade.

Print in the console:

- **`"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
5.32	Passed!

Input	Output
2.34	*(no output)*

Solution

We need to take as an input a floating-point number from the console. We will use **`double.Parse()`** to convert **string** to **double**, which we receive from **`Console.ReadLine()`**. After that, we compare the grade with **3.00** and print the result **only if** the condition returns **true**.

```
var grade = double.Parse(Console.ReadLine());
if (grade >= 3.00)
{
    Console.WriteLine("Passed!");
}
```

3. Passed or Failed

Modify the program from the previous problem, 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
5.32	Passed!

Input	Output
2.36	Failed!

Solution

We start by reading a **floating-point** number from the console. Next, we print in the **`else`** statement the appropriate message.

```
var grade = [illegible]
if (grade >= 3.00)
{
    Console.WriteLine("Passed!");
}
else
{
    [illegible]
}
```

4. Back in 30 Minutes

Every time John tries to pay the bills he sees on the cash desk the sign: **`"I will be back in 30 minutes"`**. One day John was tired of waiting and decided he needed a program, which **prints the time** after **30 minutes**, so he could come back after exactly **30 minutes**. He is not able to write the program by himself, so he asks for help.

Input

Two numbers are read from the console:

- **The first number** is **hours** and will be between **0 and 23**.
- **The second number** is **minutes** and will be between **0 and 59**.

Output

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

Examples

Input	Output
1 46	2:16

Input	Output
0 01	0:31

Input	Output
23 59	0:29

Input	Output
11 08	11:38

Input	Output
11 32	12:02

Hints

- Add 30 minutes to the initial minutes, which 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.

5. Month Printer

Create a program that receives an **integer** and prints the matching **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
2	February

Input	Output
13	Error!

Solution

```
var day = int.Parse(Console.ReadLine());
switch (day)
{
    case 1:
        Console.WriteLine("January");
        break;
    case 2:
        Console.WriteLine("February");
        break;
    // Add the rest of the cases
    case 12:
        Console.WriteLine("December");
        break;
    default:
        Console.WriteLine("Error!");
        break;
}
```

6. Foreign Languages

Create a program that prints the spoken language in a country. You will receive only the following combinations:

- English **is spoken** in England and the USA.
- Spanish **is spoken** in Spain, Argentina, and Mexico.
- For the others**,** we should print **"unknown"**.

Input

You will receive a single line of input, representing the **country name**.

Output

Print the **language** that is spoken in the given country. In case the country is **unknown** for the program, print **"unknown"**.

Examples

Input	Output
USA	English

Input	Output
Germany	unknown

Hint

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

7. Theatre Promotions

A theatre **sails tickets at discount** and a program is needed to **calculate** the price of a single ticket. If the given age does not fit one of the categories**,** you should print **"Error!"**.

The prices of the tickets are as follows:

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** – the **age** of the person.

Output

Print the price of the ticket 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$

Solution

Step 1. Read the Input

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 "`ToLower()`". After that, we will read the **age** of the person and declare a **variable** – `price`, which we will use to set the price of the ticket.

```
var day = Console.ReadLine().ToLower();
var age = int.Parse(Console.ReadLine());
var price = 0;
```

Step 2. Add If-else Statements for the Different Types of Day

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

```
if (day == "weekday")
{
    if ((age >= 0 && age <= 18) || (age > 64 && age <= 122))
    {
        price = 12;
    }
    else if (age > 18 && age <= 64)
    {
        price = 18;
    }
}
// Add the other cases
```

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

Step 3. Print the Result

We can check if the price has a value different than the initial one. If It does, that means we got a valid combination of day and age, and the price of the ticket 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.

```
if (price != 0)
{
    [redacted]
}
else
{
    [redacted]
}
```

8. Divisible by 3

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

Solution

```
for (int i = 3; i <= 100; i += 3)
{
    Console.WriteLine(i);
}
```

9. Sum of Odd Numbers

Create a program that prints on a new line the next **n odd numbers** (starting from 1). On the **last row** print the **sum of all n odd numbers**.

Input

A single number n is read from the console, indicating how many odd numbers need to be printed.

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
5	1 3 5 7 9 Sum: 25

Input	Output
3	1 3 5 Sum: 9

Solution

```
var n = int.Parse(Console.ReadLine());
var sum = 0;

for (var i = 1; i <= n; i++)
{
    [illegible]
}
Console.WriteLine([illegible]);
```

10. Multiplication Table

Create a program that receives an **integer** as an input. 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
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

Input	Output
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

11. Multiplication Table 2.0

Rewrite the program from the previous task so it can receive the **multiplier from the console**. Print the **table with the multiplier** in the interval from **the given number to 10**. If the given multiplier is **more than 10**, print only one row with the **integer**, the given **multiplier,** and the **product**. 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
5 1	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

Input	Output
2 5	2 X 5 = 10 2 X 6 = 12 2 X 7 = 14 2 X 8 = 16 2 X 9 = 18 2 X 10 = 20

Input	Output
2 14	2 X 14 = 28

12. Even Number

Create a program that reads a sequence of numbers. If the number is even, **print its absolute value in the following format: "`The number is: {number}`"** and **terminate** the program**.** If the number is odd, print "**`Please write an even number.`**" and continue reading numbers.

Examples

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

Input	Output
-6	The number is: 6

13. Debug the Code: Holidays Between Two Dates

You are assigned to find and fix all bugs in the existing code. By using the Visual Studio debugger, place a breakpoint and find the lines of code that produce incorrect or unexpected results.

You are given a program (existing **source code**) that aims to **count the non-working days between two dates** in format **`day.month.year`** (e.g. between **1.05.2015** and **15.05.2015** there are **5** non-working days – Saturday and Sunday).

Examples

Input	Output	Comments
1.05.2016 15.05.2016	5	There are 5 non-working days (Saturday / Sunday) in this period: 1-May-2016, 7-May-2016, 8-May-2016, 14-May-2016, 15-May-2016
1.5.2016 2.5.2016	1	Only 1 non-working day in the specified period: 1.05.2016 (Sunday)
15.5.2020 10.5.2020	0	The second date is before the first. No dates in the range.
22.2.2020 1.3.2020	4	Two Saturdays and Sundays: • 22.02.2020 and 23.02.2020 • 29.02.2020 and 1.03.2020

You can **find the broken code** in the judge system: Broken Code for Refactoring. It looks as follows:

HolidaysBetweenTwoDates.cs

```
using System;
using System.Globalization;

class HolidaysBetweenTwoDates
{
    static void Main()
    {
        var startDate = DateTime.ParseExact(Console.ReadLine(),
            "dd.m.yyyy", CultureInfo.InvariantCulture);
        var endDate = DateTime.ParseExact(Console.ReadLine(),
            "dd.m.yyyy", CultureInfo.InvariantCulture);
        var holidaysCount = 0;
        for (var date = startDate; date <= endDate; date.AddDays(1))
            if (date.DayOfWeek == DayOfWeek.Saturday &&
                date.DayOfWeek == DayOfWeek.Sunday) holidaysCount++;
        Console.WriteLine(holidaysCount);
    }
}
```

Hints

There are **4 mistakes** in the code. You've got to **use the debugger** to find them and fix them. After you do that, submit your **fixed code in the judge contest**.