LAB: INTRO AND BASIC SYNTAX

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`

```
using System;

namespace _01_StudentInformation
{
   internal class Program
   {
      static void Main(string[] args)
      {
         string studentName = Console.ReadLine();
         int studentAge=int.Parse(Console.ReadLine());
         double studentGrade=double.Parse(Console.ReadLine());
         Console.WriteLine($"Name: {studentName}, Age: {studentAge}, Grade: {studentGrade:f2}");
      }
   }
}
```

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

Input	Output

5.32	Passed!

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!");
}
```

```
using System;
namespace _02_Passed
{
   internal class Program
   {
      static void Main(string[] args)
      {
        double 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]
}
```

```
using System;
namespace _03_PassedOrFailed
{
   internal class Program
   {
      static void Main(string[] args)
      {
         double grade = double.Parse(Console.ReadLine());
         if (grade >= 3.00)
         {
            Console.WriteLine("Passed!");
         }
         else
         {
            Console.WriteLine("Failed!");
         }
      }
   }
}
```

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.

SOLUTION

```
using System;

namespace _04_BackIn30Minutes

{

   internal class Program

   {

      static void Main(string[] args)

      {

         int hours = int.Parse(Console.ReadLine());

         int minutes = int.Parse(Console.ReadLine());

         minutes += 30;

         if (minutes >= 60)

         {

            minutes -= 60;

            hours += 1;

         }

         if (hours >= 24)

         {

            hours -= 24;

         }

         Console.WriteLine($"{hours}:{minutes:D2}");

      }

   }

}
```

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;
}
```

```
using System;
namespace _05_MonthPrinter
{
   internal class Program
   {
      static void Main(string[] args)
      {
         int month = int.Parse(Console.ReadLine());
         switch (month)
         {
            case 1: Console.WriteLine("January"); break;
            case 2: Console.WriteLine("February"); break;
            case 3: Console.WriteLine("March"); break;
            case 4: Console.WriteLine("April"); break;
            case 5: Console.WriteLine("May"); break;
            case 6: Console.WriteLine("June"); break;
            case 7: Console.WriteLine("July"); break;
            case 8: Console.WriteLine("August"); break;
            case 9: Console.WriteLine("September"); break;
            case 10: Console.WriteLine("October"); break;
            case 11: Console.WriteLine("November"); break;
            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.

```
using System;
namespace _06_ForeignLanguages
{
  internal class Program
  {
    static void Main(string[] args)
    {
      string country = Console.ReadLine();
      switch (country)
      {
        case "USA":
        case "England":
          Console.WriteLine("English");
          break;
        case "Spain":
        case "Argentina":
        case "Mexico":
          Console.WriteLine("Spanish");
          break;
        default:
          Console.WriteLine("unknown");
          break;
```

```
        }
    }
  }
}
```

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
Weekday 42	18$

Input	Output
Holiday -12	Error!

Input	Output
Holiday 15	5$

Input	Output
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)
{
    [illegible]
}
else
{
    [illegible]
}
```

```
using System;
namespace _07_TheatrePromotion
{
  internal class Program
  {
    static void Main(string[] args)
    {
      string dayType = Console.ReadLine();
      int age = int.Parse(Console.ReadLine());
      int ticketPrice = 0;
      if (age < 0 || age > 122)
      {
        Console.WriteLine("Error!");
        return;
      }
      switch (dayType)
      {
        case "Weekday":
          if (age >= 0 && age <= 18)
            ticketPrice = 12;
          else if (age > 18 && age <= 64)
            ticketPrice = 18;
          else if (age > 64 && age <= 122)
            ticketPrice = 12;
          break;
```

```
            case "Weekend":
                if (age >= 0 && age <= 18)
                    ticketPrice = 15;
                else if (age > 18 && age <= 64)
                    ticketPrice = 20;
                else if (age > 64 && age <= 122)
                    ticketPrice = 15;
                break;
            case "Holiday":
                if (age >= 0 && age <= 18)
                    ticketPrice = 5;
                else if (age > 18 && age <= 64)
                    ticketPrice = 12;
                else if (age > 64 && age <= 122)
                    ticketPrice = 10;
                break;
            default:
                Console.WriteLine("Error!");
                return;
        }
        Console.WriteLine($"{ticketPrice}$");
    }
  }
}
```

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);
}
```

```
using System;
namespace _08_DivisibleBy3
{
  internal class Program
  {
    static void Main(string[] args)
    {
      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]
    [illegible]
}
Console.WriteLine([illegible]);
```

```
using System;
namespace _09_SumOfOddNumbers
{
  internal class Program
  {
    static void Main(string[] args)
    {
      int n = int.Parse(Console.ReadLine());
      int row = n;
      int sum = 0;
      for (int i = 1; i <= row; i++)
      {

        if (i % 2 != 0 ) {
          row++;
          sum+=i;
          Console.WriteLine(i);
```

```
            }

        }
        Console.WriteLine($"Sum: {sum}");
    }
  }
}
```

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

```
using System;
namespace _10_MultiplicationTable
{
  internal class Program
  {
    static void Main(string[] args)
    {
      int n = int .Parse(Console.ReadLine());
      for (int i = 1; i <= 10; i++)
      {
        Console.WriteLine($"{n} X {i} = {n*i}");
      }
    }
  }
}
```

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

```
using System;
namespace _12_EvenNumber
{
   internal class Program
   {
      static void Main(string[] args)
      {
         while (true)
         {
            int number = int.Parse(Console.ReadLine());
            if (number % 2 != 0)
            {
               Console.WriteLine("Please write an even number.");
            }
            else
            {
               Console.WriteLine($"The number is: {Math.Abs(number)}");
```

```
            break;
          }
        }
      }
    }
  }
}
```

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.

```
using System;
using System.Globalization;
namespace _13_HolidaysBetweenTwoDates
{
   internal class Program
   {
      static void Main(string[] args)
      {
         var startDate = DateTime.ParseExact(Console.ReadLine(),"d.M.yyyy", CultureInfo.InvariantCulture);
         var endDate = DateTime.ParseExact(Console.ReadLine(), "d.M.yyyy", CultureInfo.InvariantCulture);
         var holidaysCount = 0;
         for (var i = startDate; i <= endDate; i=i.AddDays(1))
         {
            if (i.DayOfWeek == DayOfWeek.Saturday || i.DayOfWeek == DayOfWeek.Sunday)
            {
               holidaysCount++;
               startDate.AddDays(1);
            }

         }

         Console.WriteLine(holidaysCount);
      }
   }
}
```

EXERCISE: INTRO AND BASIC SYNTAX

- You can check your solutions in Judge
- Ask your questions here https://www.slido.com/ by entering the code **#fund-csharp**

1. AGES

Write a program that determines if a person is **baby**, **child**, **teenager**, **adult** or **elder** based on the given age. The boundaries are:

- **0-2 – baby**
- **3-13 – child**
- **14-19 – teenager**
- **20-65 – adult**
- **>= 66 – elder**

All the values are **inclusive**.

EXAMPLES

Input	Output
20	adult
1	baby
100	elder

```
using System;

namespace _01_Ages
{
   internal class Program
   {
      static void Main(string[] args)
      {
         int ages = int.Parse(Console.ReadLine());
         if ( ages <= 2)
         {
            Console.WriteLine("baby");
         }
         else if(ages <= 13)
         {
             Console.WriteLine("child");
         }
         else if ( ages <= 19)
         {
            Console.WriteLine("teenager");
         }
         else if (ages <= 65)
         {
            Console.WriteLine("adult");
         }
         else if(ages >= 66)
            {
            Console.WriteLine("elder");
         }
      }
   }
}
```

2. DIVISON

You will be given an integer, write a program which checks if the given integer is divisible by **2** or **3**, or **6**, or **7**, or **10** without a remainder. You should **always take the bigger division**:

- If the number is divisible by both **2**, **3**, and **6**, you should print the **division by 6 only**.
- If a number is divisible by **2** and **10**, you should print the **division by 10**.

If the number is not divisible by any of the given numbers, print **"Not divisible".** Otherwise, print **"The number is divisible by {number}"**.

EXAMPLES

Input	Output
30	The number is divisible by 10
15	The number is divisible by 3
12	The number is divisible by 6

1643	Not divisible

```
using System;
namespace _02_Divison
{
   internal class Program
   {
      static void Main(string[] args)
      {
         int number = int.Parse(Console.ReadLine());
         if (number % 10 == 0)
         {
            Console.WriteLine($"The number is divisible by 10");
         }
         else if (number % 7 == 0)
         {
            Console.WriteLine($"The number is divisible by 7");
         }
         else if (number % 6 == 0)
         {
            Console.WriteLine($"The number is divisible by 6");
         }
         else if (number % 3 == 0)
         {
            Console.WriteLine($"The number is divisible by 3");
         }
         else if (number % 2 == 0)
         {
            Console.WriteLine($"The number is divisible by 2");
         }
         else
         {
            Console.WriteLine("Not divisible");
         }
      }
   }
}
```

3. VACATION

You will receive three lines from the console:

- A **count of people** who are going on vacation.
- **Type** of the group (**`Students`, `Business` or `Regular`**).
- The **day** of the week which the group will stay on (**`Friday`, `Saturday`** or **`Sunday`**).

Based on the given information calculate how much the group will pay for the entire vacation.

The price for a **single person** is as follows:

	Friday	Saturday	Sunday
Students	8.45	9.80	10.46

Business	10.90	15.60	16
Regular	15	20	22.50

There are also discounts based on some conditions:

- For **`Students`** – if the group is 30 or more people, you should reduce the **total** price by **15%.**
- For **`Business`** – if the group is 100 or more people, **10** of the people stay **for free.**
- For **`Regular`** – if the group is between 10 and 20 people (both inclusively), reduce the **total** price by **5%.**

Note: You should reduce the prices in that EXACT order!

As an output print the final price which the group is going to pay in the format:

`"Total price: {price}"`

The price should be **formatted to the second decimal point**.

EXAMPLES

Input	Output

`30`
`Students`
`Sunday`
	`Total price: 266.73`

`40`
`Regular`
`Saturday`
	`Total price: 800.00`

```
using System;
namespace _03_Vacation
{
   internal class Program
   {
      static void Main(string[] args)
      {
         int countOfPeople= int.Parse(Console.ReadLine());
         string typeOfGroup = Console.ReadLine();
         string day = Console.ReadLine();
         double pay = 0.0;
         //output
         switch (day)
         {
            case "Friday":
               switch (typeOfGroup)
               {
                  case "Students": pay = 8.45; break;
                  case "Business": pay = 10.90; break;
                  case "Regular":  pay = 15.00; break;
               }
               break;
            case "Saturday":
               switch (typeOfGroup)
               {
                  case "Students": pay = 9.80; break;
                  case "Business": pay = 15.60; break;
                  case "Regular":  pay = 20.00; break;
```

```
        }
        break;
    case "Sunday":
        switch (typeOfGroup)
        {
            case "Students": pay = 10.46; break;
            case "Business": pay = 16.00; break;
            case "Regular":  pay = 22.50; break;
        }
        break;
}
// Console.WriteLine($"pay: {pay}");              //int bill = 0;
if (typeOfGroup == "Students")
{
    if(countOfPeople>=30)
    {
        double bill = countOfPeople*pay;
        double totalBill = bill - (bill * 0.15);
        Console.WriteLine($"Total price: {totalBill:f2}");
    }
    else
    {
    double totalBill = countOfPeople * pay;
    Console.WriteLine($"Total price: {totalBill:f2}");
    }
}
 if (typeOfGroup == "Business")
{
    if(countOfPeople >= 100)
    {                       // double bill = (countOfPeople-10) * pay;
        double totalBill = (countOfPeople - 10) * pay;
        Console.WriteLine($"Total price: {totalBill:f2}");
    }
    else
    {
         double totalBill = countOfPeople * pay;
         Console.WriteLine($"Total price: {totalBill:f2}");
    }
}
 if (typeOfGroup == "Regular")
{
    if (countOfPeople >= 10 && countOfPeople <= 20)
    {
        double bill = countOfPeople * pay;
        double totalBill = bill - (bill * 0.05);
        Console.WriteLine($"Total price: {totalBill:f2}");
    }
    else
    {
        double totalBill = countOfPeople * pay;
        Console.WriteLine($"Total price: {totalBill:f2}");
    }
```

```
        }
    }
  }
}
```

4. PRINT AND SUM

You will receive two whole numbers from the console representing the **start** and the **end** of a **sequence of numbers**.

Your task is to print two lines:

- On the **first line, print** all numbers from the **start** of the sequence to the **end inclusive**.
- On the second line, print the sum of the sequence in the format: **`"Sum: {sum}"`**.

EXAMPLES

Input	Output
5 10	5 6 7 8 9 10 Sum: 45
0 26	0 1 2 … 26 Sum: 351
50 60	50 51 52 53 54 55 56 57 58 59 60 Sum: 605

```
using System;
namespace _04_PrintAndSum
{
  internal class Program
  {
    static void Main(string[] args)
    {
      int firstNum = int.Parse(Console.ReadLine());
      int secondNum = int.Parse(Console.ReadLine());
      int sum=0;
      for (int i = firstNum; i <= secondNum; i++)
      {
        Console.Write($"{i} ");
        sum+=i;
      }
      Console.WriteLine();
      Console.WriteLine($"Sum: {sum}");
    }
  }
}
```

5. LOGIN

On the first line, you will be given a username and your task is to try to **log in the user**. The user's password is the **username reversed**. On the next lines, you will receive passwords:

- If the password is incorrect, print **`"Incorrect password. Try again."`**.
- If the password is correct, print **`"User {username} logged in."`** and stop the program.

Keep in mind that if the password is still incorrect on the fourth attempt, you should print: **"User {username} blocked!"**.

Then the program stops.

EXAMPLES

Input	Output
Acer login go let me in recA	Incorrect password. Try again. Incorrect password. Try again. Incorrect password. Try again. User Acer logged in.
momo omom	User momo logged in.
sunny rainy cloudy sunny not sunny	Incorrect password. Try again. Incorrect password. Try again. Incorrect password. Try again. User sunny blocked!

```
using System;
namespace _05_Login
{
   internal class Program
   {
      static void Main(string[] args)
      {
         string username = Console.ReadLine();
         string pass = string.Empty;
         for (int i = username.Length - 1; i >= 0; i--)
         {
            char currChar = username[i];
            pass += currChar;
         }
         int counter = 1;
         while (counter < 5)
         {
            string inputPass = Console.ReadLine();
            if (inputPass == pass)
            {
               Console.WriteLine($"User {username} logged in.");
               break;
            }
            else
            {
               if (counter == 4)
               {
                  Console.WriteLine($"User {username} blocked!");
                  break;
               }
               if (counter < 4)
               {
                  Console.WriteLine("Incorrect password. Try again."); counter++;
               }
            }
```

```
        }
      }
    }
}
```

6. STRONG NUMBER

Write a program that receives a single **integer** and calculates if it's **strong** or **not**. A number is strong, if the **sum of the factorials** of each digit is equal to the number itself.

Example: 145 is a strong number, because **1! + 4! + 5! = 145.**

Print "**yes**", if the number is strong and "**no**", if the number is not strong.

EXAMPLES

Input	Output
2	yes
3451	no
40585	yes

```
using System;
namespace _06_StrongNumber
{
   internal class Program
   {
      static void Main(string[] args)
      {
         int number = int.Parse(Console.ReadLine());
         int originalNum = number;
         int sum = 0;
         while (number > 0)
         {
            int fac = 1;
            int lastDigth = number % 10;
            number /= 10;
            for (int i = 1; i <= lastDigth; i++)
            {
               fac *= i;
            }
            sum += fac;
         }
         if (sum == originalNum)
         {
            Console.WriteLine("yes");
         }
         else
         {
            Console.WriteLine("no");
         }
      }
```

```
    }
}
```

7. VENDING MACHINE

Write a program that accumulates coins. Until the "**Start**" command is given, you will receive coins, and only the **valid ones should be** accumulated**. Valid coins are:**

- **0.1**, **0.2**, **0.5, 1 and 2**

If an invalid coin is inserted, print "**Cannot accept {money}**" and continue to the next line.

On the next lines, until the "**End**" command is given, you will start receiving products, which a customer wants to buy. **The vending machine has only:**

- "**Nuts**" with a price of **2.0**
- "**Water**" with a price of **0.7**
- "**Crisps**" with a price of **1.5**
- "**Soda**" with a price of **0.8**
- "**Coke**" with a price of **1.0**

If the customer tries to purchase a not existing product, print "**Invalid product**".

When a customer has enough money to buy the selected product, print "**Purchased {product name}**", otherwise print "**Sorry, not enough money**" and continue to the next line.

When the "**End**" command is given print the reminding balance, formatted to the second decimal point: "**Change: {money left}**".

EXAMPLES

Input	Output
1 1 0.5 0.6 Start Coke Soda Crisps End	Cannot accept 0.6 Purchased coke Purchased soda Sorry, not enough money Change: 0.70
1 Start Nuts Coke End	Sorry, not enough money Purchased coke Change: 0.00

```
using System;
namespace _07.Vending_Machine
{
   class Program
   {
      static void Main(string[] args)
      {
         var input = Console.ReadLine();
         var sumOfMoney = 0.0;
```

```
while (input != "Start")
{
   var money = double.Parse(input);
   if (money != 0.1 && money != 0.2 && money != 0.5 && money != 1 && money != 2)
   {
      Console.WriteLine($"Cannot accept {money}");
   }
   else
   {
      sumOfMoney += money;
   }
   input = Console.ReadLine();
}
var totalMoney = sumOfMoney;
var change = 0.0;
while (input != "End")
{
   input = Console.ReadLine();
   var snacks = input;
   switch (snacks)
   {
      default:
         if (snacks != "End")
         {
            Console.WriteLine("Invalid product");
         }
         continue;
      case "Nuts":
         if (sumOfMoney >= 2.0)
         {
            sumOfMoney -= 2.0;
            Console.WriteLine("Purchased nuts");
         }
         else
         {
            Console.WriteLine("Sorry, not enough money");
            change = sumOfMoney;
         }
         break;
      case "Water":
         if (sumOfMoney >= 0.7)
         {
            sumOfMoney -= 0.7;
            Console.WriteLine("Purchased water");
         }
         else
         {
            Console.WriteLine("Sorry, not enough money");
            change = sumOfMoney;
         }
         break;
      case "Crisps":
```

```
                if (sumOfMoney >= 1.5)
                {
                    sumOfMoney -= 1.5;
                    Console.WriteLine("Purchased crisps");
                }
                else
                {
                    Console.WriteLine("Sorry, not enough money");
                    change = sumOfMoney;
                }
                break;
            case "Soda":
                if (sumOfMoney >= 0.8)
                {
                    sumOfMoney -= 0.8;
                    Console.WriteLine("Purchased soda");
                }
                else
                {
                    Console.WriteLine("Sorry, not enough money");
                    change = sumOfMoney;
                }
                break;
            case "Coke":
                if (sumOfMoney >= 1.0)
                {
                    sumOfMoney -= 1.0;
                    Console.WriteLine("Purchased coke");
                }
                else
                {
                    Console.WriteLine("Sorry, not enough money");
                    change = sumOfMoney;
                }
                break;
        }
    }
    if (sumOfMoney > 0)
    {
        change = sumOfMoney;
        Console.WriteLine($"Change: {change:f2}");
    }
    else
    {
        change = Math.Abs(sumOfMoney);
        Console.WriteLine($"Change: {change:f2}");
    }
  }
}
```

8. TRIANGLE OF NUMBERS

Write a program, which receives a number – **n** and prints a triangle from **1 to n**.

CONSTRAINTS

- **n** will be in the interval [**1...20**].

EXAMPLES

Input	Output
3	1 2 2 3 3 3

Input	Output
5	1 2 2 3 3 3 4 4 4 4 5 5 5 5 5

Input	Output
6	1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 6 6 6 6 6 6

```
using System;
namespace _08_TriangleOfNumbers
{
   internal class Program
   {
      static void Main(string[] args)
      {
         int num = int.Parse(Console.ReadLine());
         //int row = 1;
         for (int row = 1; row <= num; row++)
         {

            for (int col = 1; col <= row; col++)
            {
               Console.Write(row + " ");
            }
            Console.WriteLine();
         }
      }
   }
}
```

9. *Padawan Equipment

Yoda is starting his newly created Jedi academy. So, he asked master John to **buy** the **needed equipment**. The number of **items** depends on **how many students will sign up**. The equipment for each Padawan contains:

- **Lightsaber**
- **Belt**
- **Robe**

You will be given **the amount of money John has**, the **number of students** and the **prices of each item**. Calculate if John has enough **money to buy equipment for each Padawan** or how much more money he needs.

There are some additional requirements:

- Lightsabres sometimes break, so John should **buy 10% more (taken from the students' count)**, **rounded up** to the next integer.

- Every **sixth belt is free.**

INPUT / CONSTRAINTS

The input data should be read from the console. It will consist of **exactly 5 lines**:

- The **amount of money** John has – **floating-point number** in the **range [0.00…1000.00].**
- The **count of students – integer in the range [0…100].**
- The **price of lightsabers** for a **single saber – floating-point number** in the **range [0.00…100.00].**
- The **price of robes** for a **single robe – floating-point number** in the **range [0.00…100.00].**
- The **price of belts** for a **single belt – floating-point number** in the **range [0.00…100.00].**

The **input data will always be valid**. **There is no need to check it explicitly**.

OUTPUT

The output should be printed on the console.

- If the calculated price of the equipment is less or equal to the money John has:
  - **"The money is enough - it would cost {the cost of the equipment}lv."**
- If the calculated price of the equipment is more than the money John has:
  - **" John will need {neededMoney}lv more."**
- All prices must be rounded to two digits after the decimal point.

EXAMPLES

Input	Output	Comments
100 2 1.0 2.0 3.0	The money is enough - it would cost 13.00lv.	Needed equipment for 2 padawans : **sabresPrice** * (**studentsCount** + 10%) + **robesPrice** * (**studentsCount**) + **beltsPrice** * (**studentsCount** – **freeBelts**) 1*(3) + 2*(2) + 3*(2) = 13.00 13.00 <= 100 – the money will be enough.
Input	**Output**	**Comments**
100 42 12.0 4.0 3.0	John will need 737.00lv more.	Needed equipment for 42 padawans: 12 * 47 + 4 * 42 + 3 * 35 = 837.00 837 > 100 – need 737.00 lv. more.

```
using System;
namespace _09_PadawanEquipment
{
  internal class Program
  {
    static void Main(string[] args)
    {
      double amount = double.Parse(Console.ReadLine());
      int studentsCount = int.Parse(Console.ReadLine());
      double sabresPrice = double.Parse(Console.ReadLine());
      double robesPrice = double.Parse(Console.ReadLine());
      double beltsPrice = double.Parse(Console.ReadLine());
      int freeBelts = studentsCount - (studentsCount / 6);
      double sabresCount = Math.Ceiling(studentsCount * 1.1);
```

```
        double totalMoney = sabresPrice * sabresCount + robesPrice * studentsCount + beltsPrice * freeBelts;

        if (totalMoney <= amount)
        {
            Console.WriteLine($"The money is enough - it would cost {(totalMoney):f2}lv.");
        }
        else
        {
            Console.WriteLine($" John will need {Math.Abs(amount - totalMoney):f2}lv more.");
        }
    }
  }
}
```

10. *RAGE EXPENSES

As a MOBA challenger player, Petar has the bad habit of trashing his PC when he loses a game and of rage quiting. His gaming setup consists of a **headset, mouse, keyboard, and display**. You will receive Petar's **lost games count**.

Every **second** lost game, Petar trashes his **headset.**

Every **third** lost game, Petar trashes his **mouse**.

When Petar trashes **both his mouse and headset** in the **same** lost game, he also trashes his **keyboard**.

Every second time, when he trashes his keyboard, he also trashes his **display**.

You will receive the price of each item in his gaming setup. Calculate his rage expenses for renewing his gaming equipment.

INPUT / CONSTRAINTS

- On the first input line – **lost games count** – integer in the range **[0…1000].**
- On the second line – **headset price** – floating-point number in the range **[0…1000].**
- On the third line – **mouse price** – floating-point number in the range **[0…1000].**
- On the fourth line – **keyboard price** – floating-point number in the range **[0…1000].**
- On the fifth line – **display price** – floating-point number in the range **[0… 1000].**

OUTPUT

- As output you must print Petar's total expenses: **`"Rage expenses: {expenses} lv."`.**
- Allowed working **time** / **memory**: **100ms** / **16MB**.

EXAMPLES

Input	Output	Comment
7 2 3 4 5	`Rage expenses: 16.00 lv.`	Trashed headset → 3 times Trashed mouse → 2 times Trashed keyboard → 1 time Total: 6 + 6 + 4 = 16.00 lv

23 12.50 21.50 40 200	Rage expenses: 608.00 lv.	

```
using System;
namespace _10_RageExpenses
{
   internal class Program
   {
      static void Main(string[] args)
      {
         int lostGame = int.Parse(Console.ReadLine());
         decimal headsetPrice = decimal.Parse(Console.ReadLine());
         decimal mousePrice = decimal.Parse(Console.ReadLine());
         decimal keyboardPrice = decimal.Parse(Console.ReadLine());
         decimal displayPrice = decimal.Parse(Console.ReadLine());
         int headsetCount = 0;
         int mouseCount = 0;
         int keyboardCount = 0;
         int displayCount = 0;
         for (int i = 1; i <= lostGame; i++)
         {
            if (i % 2 == 0)
            {
               headsetCount++;
            }
            if (i % 3 == 0)
            {
               mouseCount++;
            }
            if (i % 2 == 0 && i % 3 == 0)
            {
               keyboardCount++;
               if (keyboardCount % 2 == 0)
               {
                  displayCount++;
               }
            }
         }
          decimal expenses = headsetCount * headsetPrice + mouseCount * mousePrice + keyboardCount * keyboardPrice +
displayCount * displayPrice;
         Console.WriteLine($"Rage expenses: {expenses:f2} lv.");
      }
   }
}
```

11. *ORDERS

We are placing **N** orders at a time. You need to calculate the price with the following formula:

```
((daysInMonth * capsulesCount) * pricePerCapsule)
```

INPUT / CONSTRAINTS

- On the first line, you will receive integer **N** – the count of orders the shop will receive.
- For each order you will receive the following information:
  - Price per capsule – **floating-point number** in the range **[0.00…1000.00].**
  - Days – **integer** in the range **[1…31].**
  - Capsules count – **integer** in the range **[0…2000].**

The input will be in the described format, there is no need to check it explicitly.

OUTPUT

The output should consist of **N + 1** line. For each order you must print a single line in the following format:

- **"The price for the coffee is: ${price}"**

On the last line, you need to print the total price in the following format:

- **"Total: ${totalPrice}"**

The **price must be formatted** to 2 decimal places.

EXAMPLES

Input	Output	Comments
1 1.53 30 8	The price for the coffee is: $367.20 Total: $367.20	We are given only 1 order. Then we use the formula: **orderPrice** = 30 * 8 * 1.53 = 367.20
2 4.99 31 3 0.35 31 5	The price for the coffee is: $464.07 The price for the coffee is: $54.25 Total: $518.32	
1 9.223 31 433	The price for the coffee is: $123800.33 Total: $123800.33	

```
using System;
namespace _11_Orders
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int order = int.Parse(Console.ReadLine());
            decimal totalPrice = 0;

            for (int i = 0; i < order; i++)
            {
```

```
            decimal pricePerCapsule = decimal.Parse(Console.ReadLine());
            int  daysInMonth = int.Parse(Console.ReadLine());
           int  capsulesCount = int.Parse(Console.ReadLine());
            decimal price = ((daysInMonth * capsulesCount) * pricePerCapsule);
             Console.WriteLine($"The price for the coffee is: ${price:f2}");
             totalPrice += price;
        }                       // Console.WriteLine($"The price for the coffee is: ${price}");
        Console.WriteLine($"Total: ${totalPrice:f2}");
    }
  }
}
```

MORE EXERCISE: INTRO AND BASIC SYNTAX

You can check your solutions in Judge

1. SORT NUMBERS

Create a program that receives three real numbers and sorts them in descending order. Print each number on a new line.

EXAMPLES

Input	Output
2 1 3	3 2 1
-2 1 3	3 1 -2
0 0 2	2 0 0

```
using System;
namespace _01_SortNumbers
{
  internal class Program
  {
    static void Main(string[] args)
    {
      int n1 = int.Parse(Console.ReadLine());
      int n2 = int.Parse(Console.ReadLine());
      int n3 = int.Parse(Console.ReadLine());
      if (n1> n2 && n1 > n3)
      {
        if (n2 > n3)
        {
          Console.WriteLine($"{n1}\n{n2}\n{n3}");
        }
        else
        {
          Console.WriteLine($"{n1}\n{n3}\n{n2}");
        }
```

```
        }
        else if (n2 > n1 && n2 > n3)
        {
            if (n1 > n3)
            {
                Console.WriteLine($"{n2}\n{n1}\n{n3}");
            }
            else
            {
                Console.WriteLine($"{n2}\n{n3}\n{n1}");
            }
        }
        else
        {
            if (n1 > n2)
            {
                Console.WriteLine($"{n3}\n{n1}\n{n2}");
            }
            else
            {
                Console.WriteLine($"{n3}\n{n2}\n{n1}");
            }
        }
    }
  }
}
```

2. ENGLISH NAME OF THE LAST DIGIT

Create a **method** that returns the **English spelling** of the last digit of a given number. Write a program that reads an integer and prints the returned value from this method.

EXAMPLES

Input	Output
512	two
1	one
1643	three

SOLUTION

```
using System;
namespace _02_EnglishNameOfTheLastDigit
{
  internal class Program
  {
    static void Main(string[] args)
    {
      int num = int.Parse(Console.ReadLine());
      int digith = num % 10;
```

```
        switch (digith)
        {
            case 0: Console.WriteLine("zero"); break;
            case 1: Console.WriteLine("one"); break;
            case 2: Console.WriteLine("two"); break;
            case 3: Console.WriteLine("three"); break;
            case 4: Console.WriteLine("four"); break;
            case 5: Console.WriteLine("five"); break;
            case 6: Console.WriteLine("six"); break;
            case 7: Console.WriteLine("seven"); break;
            case 8: Console.WriteLine("eight"); break;
            case 9: Console.WriteLine("nine"); break;
        }
    }
  }
}
```

3. GAMING STORE

Create a program, which helps you buy the games. The **valid games** are the following games in this table:

Name	Price
OutFall 4	$39.99
CS: OG	$15.99
Zplinter Zell	$19.99
Honored 2	$59.99
RoverWatch	$29.99
RoverWatch Origins Edition	$39.99

On the first line, you will receive your **current balance** – a **floating-point** number in the range **[0.00…5000.00]**.

Until you receive the command **"Game Time"**, you have to keep **buying games**. When a **game** is **bought**, the user's **balance** decreases by the **price** of the game.

Additionally, the program should obey the following conditions:

- If a game the user is trying to buy is **not present** in the table above, print "**Not Found**" and **read the next line**.
- If at any point, the user has **$0** left, print "**Out of money!**" and **end the program**.
- Alternatively, if the user is trying to buy a game that they **can't afford**, print "**Too Expensive**" and **read the next line**.
- If the game exists and the player has the money for it, print **"Bought {nameOfGame}"**.

When you receive "**Game Time**", **print** the user's **remaining money** and **total spent on games**, **rounded** to the **2nd decimal place**.

EXAMPLES

Input	Output
120 RoverWatch Honored 2 Game Time	Bought RoverWatch Bought Honored 2 Total spent: $89.98. Remaining: $30.02
19.99	Not Found

Reimen origin RoverWatch Zplinter Zell Game Time	Too Expensive Bought Zplinter Zell Out of money!
79.99 OutFall 4 RoverWatch Origins Edition Game Time	Bought OutFall 4 Bought RoverWatch Origins Edition Total spent: $79.98. Remaining: $0.01

```
using System;
using System.Globalization;
namespace _03_GamingStore
{
   internal class Program
   {
      static void Main(string[] args)
      {
         double balance = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);
         double startBalance = balance;
         double spentMoney = 0;
         string input = "";
         bool isFound = false;
         bool isGameTime = false;
         while ((input = Console.ReadLine()) != "Game Time")
         {
            // input =Console.ReadLine();
            double price = 0;
            switch (input)
            {
               case "OutFall 4":
                  price = 39.99;
                  isFound = true;
                  break;
               case "CS: OG":
                  price = 15.99;
                  isFound = true;
                  break;
               case "Zplinter Zell":
                  isFound = true;
                  price = 19.99;
                  break;
               case "Honored 2":
                  isFound = true;
                  price = 59.99;
                  break;
               case "RoverWatch":
                  isFound = true;
                  price = 29.99;
                  break;
               case "RoverWatch Origins Edition":
                  isFound = true;
                  price = 39.99; break;
               default:
                  Console.WriteLine("Not Found");
```

```
                isFound = false;
                break;
            }
            if (balance >= price && isFound == true)
            {
                balance -= price;
                spentMoney += price;
                Console.WriteLine($"Bought {input}");
            }
            else if (balance < price && isFound == true)
            {
                Console.WriteLine("Too Expensive");
            }
            if (isGameTime == true && balance == 0)
            {
                break;
            }
        }
        if (balance != 0)
        {
            Console.WriteLine($"Total spent: ${spentMoney:f2}. Remaining: ${balance:f2}");
        }
        if (balance == 0)
        {
            Console.WriteLine("Out of money!");
            isGameTime = true;
        }
    }
  }
}
```

4. REVERSE STRING

Create a program which reverses a string and prints it on the console.

EXAMPLES

Input	Output
Hello	olleH
SoftUni	inUtfoS
1234	4321

SOLUTION

```
using System;
namespace _04_ReverseString
{
  internal class Program
  {
    static void Main(string[] args)
```

```
    {
        string input = Console.ReadLine();
        for (int i = input.Length - 1; i >= 0; i--)
        {
            Console.Write(input[i]);
        }
    }
  }
}
```

5. MESSAGES

Create a program, which emulates **typing an SMS**, following this guide:

1	**2** abc	**3** def
4 ghi	**5** jkl	**6** mno
7 pqrs	**8** tuv	**9** wxyz
	0 space	

Following the guide, **2** becomes '**a**', **22** becomes '**b**' and so on.

EXAMPLES

Input	Output
5 44 33 555 555 666	hello

Input	Output
9 44 33 999 0 8 44 33 777 33	hey there

Input	Output
7 6 33 33 8 0 6 33	meet me

HINTS

- A naive approach would be to just put all the possible combinations of digits in a giant **`switch`** statement.
- A cleverer approach would be to come up with a **mathematical formula**, which **converts** a **number** to its **alphabet** representation:

Digit	2	3	4	5	6	7	8	9
Index	0 1 2	3 4 5	6 7 8	9 11 12	13 14 15	16 17 18 19	20 21 22	23 24 25 26
Letter	a b c	d e f	g h i	j k l	m n o	p q r s	t u v	w x y z

- Let's take the number **222** (**c**) for example. Our algorithm would look like this:
  - Find the **number of digits** the number has, e.g. **222** ➔ **3 digits**
  - Find the **main digit** of the number, e.g. **222** ➔ **2**

- Find the **offset** of the number. To do that, you can use the formula: **`(main digit - 2) * 3`**
- If the main digit is **8 or 9**, you need to **add 1** to the **offset**, since the digits **7** and **9** have **4 letters each**
- Finally, find the **letter index** (a ➔ `0`, c ➔ 2, etc.). To do that, you can use the following formula: **`(offset + digit length - 1)`**.
- After you've found the **letter index**, you can just add that to **the ASCII code** of the lowercase letter '**a**' (97)

```
using System;
namespace _05_Messages
{
   internal class Program
   {
      static void Main(string[] args)
      {
         int n = int.Parse(Console.ReadLine());
         string message = string.Empty;            //  Console.WriteLine(n);
         for (int i = 0; i < n; i++)
         {
            string number = Console.ReadLine();
            //char letter = ' ';
            if (number == "0")
            {
               message += " ";
            }
            else
            {
               int firstDigit = int.Parse(number[0].ToString());       //  Console.WriteLine("first: " + firstDigit);
               int length = number.Length;                             //  Console.WriteLine("len:" + length);
               int offset = (firstDigit - 2) * 3;
               if (firstDigit == 8 || firstDigit == 9)
               {
                  offset += 1;
               }
               int index = (offset + length) + 96;       //  Console.WriteLine("index: " + index);
               char letter = (char)(index);
               message += letter;
            }
         }
         Console.WriteLine(message);
      }
   }
}
```