# Exercise: Intro and Basic Syntax

Problems for exercises and homework for the "[C# Fundamentals" course @ SoftUni](https://softuni.bg/trainings/3606/programming-fundamentals-with-csharp-january-2022)  
You can check your solutions in [Judge](https://judge.softuni.org/Contests/1204/Basic-Syntax-Conditional-Statements-and-Loops-Exercise)

## Ages

Write a program that determines if a person is **baby**, **child**, **teenager**, **adult**, **elder** based on the given age. The bounders 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 |

## Divison

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

* If the number is divisible by both **2**, **3**, and **6**, you should print only 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 |

## Vacation

You will receive three lines from the console:

* A **count of people**, which are going on vacation.
* **Type** of the group (**Students, Business, or Regular**).
* The **day** of the week which the group will stay (**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 |

## 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 |

## Login

On the first line, you will be given a username, your task is to try to **log in the user**. The user’s password is **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, on the fourth attempt if the password is still not correct 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! |

## Strong Number

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

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

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

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 | yes |
| 3451 | no |
| 40585 | yes |

## 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 |

## 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** |  | **Input** | **Output** |  | **Input** | **Output** |
| 3 | 1  2 2  3 3 3 | 5 | 1  2 2  3 3 3  4 4 4 4  5 5 5 5 5 | 6 | 1  2 2  3 3 3  4 4 4 4  5 5 5 5 5  6 6 6 6 6 6 |

## **\*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 does 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 student's 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 – the **floating-point number** in the **range [0.00…1,000.00]**
* The **count of students – integer in the range [0…100]**
* The **price of lightsabers** for a **single saber – the floating-point number** in the **range [0.00…100.00]**
* The **price of robes** for a **single robe – the floating-point number** in the **range [0.00…100.00]**
* The **price of belts** for a **single** **belt – the 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. |

*...May the force  
 be with you...*

## \*Rage Expenses

As a MOBA challenger player, Petar has the bad habit to trash his PC when he loses a game and rage quits. 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** - the floating-point number in the range **[0, 1000]**.
* On the third line – **mouse price** - the floating-point number in the range **[0, 1000]**.
* On the fourth line – **keyboard price** - the floating-point number in the range **[0, 1000]**.
* On the fifth line – **display price** - the 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. |  |

## \*Orders

We are placing **N** orders at a time. You need to calculate the price on 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 - the **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: ${**totalP**rice}"**

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 formulas:  **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 |  |