# Exercise: Intro and Basic Syntax

Problems for exercises and homework for the "[C# Fundamentals" course @ SoftUni](https://softuni.bg/trainings/3447/programming-fundamentals-with-csharp-september-2021)  
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 whether based on the given age a person is: baby, child, teenager, adult, elder. 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 and you have to print on the console whether that number is divisible by the following numbers: 2, 3, 6, 7, 10. You should **always take the bigger division**. If the number is divisible by both **2** and **3** it is also divisible by **6** and you should print only the division by 6. If a number is divisible by **2** it is sometimes also divisible by **10** and 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:

* **Students –** if the group is bigger than or equal to 30 people you should reduce the **total** price by 15%
* **Business –** if the group is bigger than or equal to 100 people **10** of them can stay **for free.**
* **Regular –** if the group is bigger than or equal to 10 and less than or equal to 20 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. The first number is the start of the sequence, the second number is the end of the sequence. 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

You will be given a username. The password will be the username reversed.

Until you receive the correct password print on the console:

"**Incorrect password. Try again.**"

When you receive the correct password print:

"**User {username} logged in.**"

However, on the fourth try 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 calculates if a given number is 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

Until you receive "**Start**" you will be given different coins that are being inserted into the machine. You have to sum them to have the total money inserted. There is a problem though.

Your vending machine only works with **0.1**, **0.2**, **0.5, 1, and 2** coins. If someone tries to insert some other coins you have to display "**Cannot accept {money}**" and **not** add it to the total money.

On the next few lines until you receive "**End**" you will be given products to purchase. Your machine has however only "**Nuts**", "**Water**", "**Crisps**", "**Soda**", "**Coke**". The prices are: **2.0**, **0.7**, **1.5**, **0.8**, **1.0** respectively. If the person tries to purchase a not existing product print "**Invalid product**".

Be careful that the person may try to purchase a product they don’t have the money for. In that case, print "**Sorry, not enough money**". If the person purchases a product successfully print "**Purchased {product name}**".

After the "**End**" command print the money that is left formatted to the second decimal point in the format "**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** as in the examples.

### 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 Ivan Cho to **buy** the **needed equipment**. The number of **items** depends on **how many students will sign up**. The equipment for the Padawan contains **lightsabers, belts, and robes**.

You will be given **the amount of money Ivan Cho has**, the **number of students,** and the **prices of each item**. You have to help Ivan Cho **calculate** if the **money** he has is **enough to buy all of the equipment**, or how much more money he needs.   
Because the lightsabres sometimes break, Ivan Cho should **buy 10% more (taken from the student's count)**, **rounded up** to the next integer. Also, 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** Ivan Cho 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 Ivan Cho 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 Ivan Cho has:**
  + "Ivan Cho 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 | Ivan Cho 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 **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 range **[0, 1000]**.
* On the third line – **mouse price** - floating point number in range **[0, 1000]**.
* On the fourth line – **keyboard price** - floating point number in range **[0, 1000]**.
* On the fifth line – **display price** - floating point number in 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. |  |

## Santa's Cookies

You need to make cookies for Santa and his dwarfs.

You will receive **the amount of batches – n** that you need to bake**. For every batch** you will receive ingredients: **flour**, **sugar** and **cocoa** in grams, each on a new line. You need to calculate how many boxes of cookies you get for every batch with the given ingredients and total boxes of cookies for all batches. To calculate the number of **boxes per batch** you need to divide **total cookies per bake** by **cookies per box** (see the table below). To get the **total cookies per bake** use the following formula and round the result to the nearest lower number:

({cup} + {smallSpoon} + {bigSpoon}) \* min from({flourCups}, {sugarSpoons}, {cocoaSpoons}) / singleCookieGrams

To get the **flourCups** divide flour by **cups**.

To get the **sugarSpoons** divide sugar by **bigSpoon**.

And for the **cocoaSpoons** divide cocoa by **smallSpoon**.

The cups and the spoons must be integer numbers.

(see the table below)

If flourCups, sugarSpoons or cocoaSpoons for a single bake are not enough (**<=0**), print the following message: "Ingredients are not enough for a box of cookies."

Otherwise calculate the cookies and print the number of boxes you get for the current batch:

"Boxes of cookies: {boxes of cookies per current bake}"

|  |  |  |
| --- | --- | --- |
| **Item** | **Grams** | |
| **Single cookie Grams** | | **25** |
| **Cup** | | **140** |
| **Small Spoon** | | **10** |
| **Big Spoon** | | **20** |
| **Cookies per Box** | | **5** |

Use the following table for calculations:

When you finish baking, pack the all the cookies in boxes and send them to Santa and his dwarfs and print the total number of boxes on the console.

**"Total boxes: {totalBoxes for all bakes }"**

### Input

The input data should be read from the console. It will consist of:

* **Amount of batches** **- integer**  **number** in **range [0…1,000,000,000]**

**For every batch:**

* **Amount of flour in grams – integer number in range [0…1,000]**
* **Amount of sugar in grams – integer number** in **range [0…1,000]**
* **Amount of cоcоа in grams – integer number** in **range [0…1,000]**

The **input data will always be valid** and in the format described. **There is no need to check it explicitly**.

### Output

The output should be printed on the console.

* **If the ingredients for current bake are not enough:**

**"Ingredients are not enough for a box of cookies."**

* **If the ingredients for current bake are enough:**

**"**Boxes of cookies: {boxes of cookies per current bake}."

* + On the last line print:

**"Total boxes: {totalBoxes for all bakes}"**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 200 300 500  100 200 50 | Boxes of cookies: 1  Ingredients are not enough for a box of cookies.  Total boxes: 1 |
| 1 1400 200 100 | Boxes of cookies: 13  Total boxes: 13 |