Exercise: JS Basic Syntax, Conditional Statements and Loops

Problems for exercise and homework for the "JS Fundamentals" Course @ SoftUni.
Submit your solutions in the SoftUni judge system at: https://judge.softuni.bg/Contests/1207

1. Ages

Write a function that determines whether based on the given age a person is: baby, child, teenager, adult, elder. The input comes as **single number parameter**. The bounders are:

- **0-2 – baby;**
- **3-13 – child;**
- **14-19 – teenager;**
- **20-65 – adult;**
- **>=66 – elder;**
- In all other cases - **out of bounds**
- All the values are **inclusive**.

Examples

Input	Output
20	adult
1	baby
100	elder

2. Rounding

Write a JS function that rounds numbers to specific precision.

The **input** comes as **two numbers**. The first value is the number to be rounded and the second is the precision (significant decimal places). If a precision is passed, that is more than **15** it should automatically be reduced to **15**.

Remove trailing zeroes, if any (you can use **parseFloat()**)

The **output** should be printed to the console. Do not print insignificant decimals.

Examples

Input	Output
3.14159265358979323846 26433832795,2	3.14

Input	Output
10.5,3	10.5

3. Division

You will be given a number and you have to return whether that number is divisible by the following numbers**: 2, 3, 6, 7**, and **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

4. Vacation

You are given a **group of people**, **type of the group**, and **day of the week** they are going to stay. Based on that information calculate how much they have to pay and print that price on the console. Use the table below. In each cell is the price for a **single person**. The output should look like that:
"**Total price: {price}**". The price should be formatted to the second decimal point.

	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 10 and less than or equal to 20 reduce the total price by 5%

You should reduce the prices in that EXACT order

Examples

Input	Output
30, "Students", "Sunday"	Total price: 266.73
40, "Regular", "Saturday"	Total price: 800.00

5. Leap Year

Write a JS function to check whether a year is leap. Leap years are either divisible by 4 but not by 100 or are divisible by 400. Return the result like examples below:

Examples

Input	Output
1984	yes
2003	no
4	yes

6. Print and Sum

Write a function to display numbers from **given start** to given **end** and their **sum**. The input comes as **two number parameters**.

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

7. Triangle of Numbers

Write a function, which receives a **single 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
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

8. Multiplication Table

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

Output

Print every row of the table in the following format:

{number} X {times} = {product}

Constraints

- The number will be an **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

9. * Login

You will be given a string representing a username. The password will be that 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!**" and end the program. The input comes as an **array of strings**.

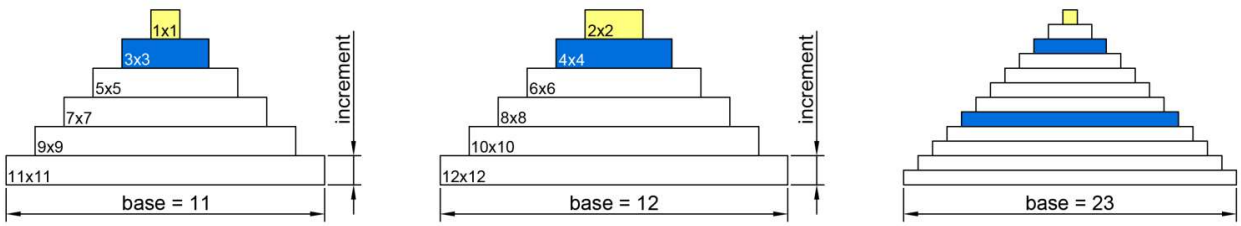
Examples

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!

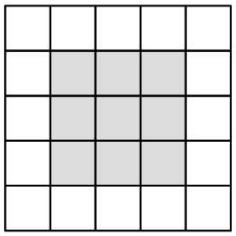
10. * the Pyramid of King Djoser

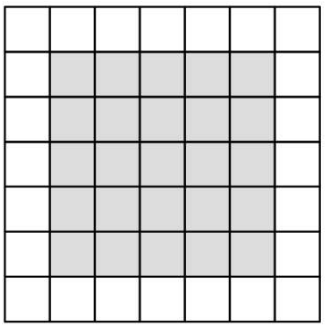
Write a JS program that calculates how much resources will be required for the construction of a pyramid. It is made out of **stone**, **marble**, **lapis lazuli** and **gold**. Your program will receive an integer that will be the **base** width and length of the pyramid and an **increment**, that is the height of each step. The bulk is made out of stone, while the **outer layer** is made out of marble. **Every fifth step's** outer layer is made out of lapis lazuli **instead** of marble. The **final step** is made out of gold.

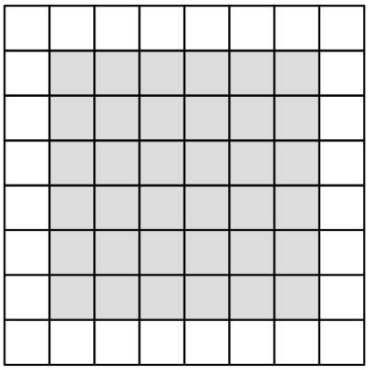
The pyramid is built with **1x1 blocks** with **height** equal to the given **increment**. The first step of the pyramid has **width** and **length** equal to the given **base** and every next step is **reduced by 2 blocks** (1 from each side). The height of every step equals the given **increment**. See the drawing for an example. White steps are covered in marble, blue steps are covered in lapis lazuli (**every fifth layer from the bottom**), and yellow steps are made **entirely** out of gold (**top-most step**).



Since the **outer layer** of each step is made of a decorative material, to calculate the required stone for one step, reduce the width and length by 2 blocks (one from each side), find it's area and multiply it by the increment. The rest of the step is made out of lapis lazuli for every fifth step from the bottom and marble for all other steps. To find the amount needed, you may, for example, find its perimeter and reduce it by 4 (to compensate for the overlapping corners) and multiply the result by the increment. See the drawing for details (grey is stone, white is decoration).







5x5 step	7x7 step	8x8 step
Stone required – 9 x increment	Stone required – 25 x increment	Stone required – 36 x increment
Marble required – 16 x increment	Marble required – 24 x increment	Marble required – 28 x increment

Note the top-most layer is made entirely out of gold, with height equal to the given increment. See the examples for complete calculations.

Input

You will receive two **number** parameters **base** and **increment**.

Output

Print on the **console** on separate lines the **total** required **amounts** of each material **rounded up** and the **final height** of the pyramid **rounded down**, as shown in the examples.

Constraints

- The **base** will always be an integer greater than zero
- The **increment** will always be a number greater than zero
- **Number.MAX_SAFE_INTEGER** will **never be exceeded** for any of the calculations

Examples

Input	Output	Explanation					
11, 1	Stone required: 165 Marble required: 112 Lapis Lazuli required: 8 Gold required: 1 Final pyramid height: 6	Step	Size	Stone	Marble	Lapis	Gold
		1st	11x11	81	40	-	-
		2nd	9x9	49	32	-	-
		3rd	7x7	25	24	-	-
		4th	5x5	9	16	-	-

		5th	3x3	1	-	8	-
		6th	1x1	-	-	-	1
		total	Height=6	165	112	8	1

Input	Output	Explanation
11, 0.75	Stone required: 124 Marble required: 84 Lapis Lazuli required: 6 Gold required: 1 Final pyramid height: 4	Total **stone** is 81*0.75+49*0.75+25*0.75+9*0.75+1*0.75 = 123.75, we round up to 124. Total **marble** is 40*0.75+32*0.75+24*0.75+16*0.75=84. Total **lapis lazuli** is 8*0.75=6. Total **gold** is 1*0.75=0.75, we round up to 1. Total **height** is 4.5 (6 **steps** times 0.75), we round down to 4.

Input	Output
12, 1	Stone required: 220 Marble required: 128 Lapis Lazuli required: 12 Gold required: 4 Final pyramid height: 6

Input	Output
23, 0.5	Stone required: 886 Marble required: 228 Lapis Lazuli required: 36 Gold required: 1 Final pyramid height: 6

11. * Bitcoin "Mining"

Write a JavaScript program that calculates the **total amount** of **bitcoins** you purchased with the gold you mined during your **shift** at the mine. Your shift consists of a certain number of days where you mine an amount of **gold** in **grams**. Your program will receive an **array with the amount of gold** you mined **each day**, where the **first day** of your **shift** is the **first index of the array**. Also, someone was stealing **every third day** from the start of your shift **30%** from the mined **gold** for **this day**. For the different exchanges use these **prices**:

1 Bitcoin	11949.16 lv.
1 g of gold	67.51 lv.

Input

You will receive an array of **numbers**, representing your **shift** at the mine.

Output

Print on the **console these lines in the following formats**:

- **First line** prints the **total amount** of bought **bitcoins**:

  "**Bought bitcoins: {count}**"
- **Second line** prints **which day** you **bought** your **first bitcoin**:

  "**Day of the first purchased bitcoin: {day}**"

  In case you **did not purchased any bitcoins,** do not print the second line.
- **Third line** prints the **amount** of **money** that's left after the bitcoin purchases **rounded by the second digit** after the decimal point:

  "**Left money: {money} lv.**"

Constraints

- The **input** array may contain up to **1,000** elements
- The numbers in the array are in range **[0.01..5,000.00] inclusive**
- Allowed time/memory: 100ms/16MB

Examples

Input	Output
[100,200,300]	Bought bitcoins: 2 Day of the first purchased bitcoin: 2 Left money: 10531.78 lv.

Scroll down to see the explanation for the first example and more examples.

Explanation
Day 1 – you dig up **100 g** of gold then exchange it for **6751.00 lv.** **Day 2** – you dig up **200 g** of gold then exchange it for **13,502.00 lv.** and the total amount of money is **20,253.00 lv.** Then you buy **1 Bitcoin** which leaves you with **8,303.84 lv.** Also this purchase is the **first day you bought bitcoin**. **Day 3** – you dig up **300 g** of gold but then **30%** of it is stolen and your gold drops to **210 g** which you exchange for **14,177.10 lv.** making your total amount of money **22,480.94 lv.** Then you buy **1 Bitcoin** making the final amount of money that you are **left with 10,531.78 lv.** with **2 bought Bitcoins.**

Input	Output
[50, 100]	Bought bitcoins: 0 Money left: 10126.50 lv.

Input	Output
[3124.15, 504.212, 2511.124]	Bought bitcoins: 30 Day of the first purchased bitcoin: 1 Money left: 5144.11 lv.