Зад 1.
Николай иска да разбере колко и какви символи се съдържат в даден низ. За низ - s, определете броя на големи букви, малки букви, числа и други символи.

Пример:
s = Hello, Pravets 2022!
На конзолата трябва да се принтира:
Digits count: 4
Lowercase letters count: 10
Uppercase letters count: 2
Other symbols count: 4

Входни параметри:
• s - String

Ограничения:
• 1 ≤ s ≤ 10000 (дължина на низа)
• Всеки символ s[i] ∈ ascii

Изходен формат:
Digits count: {number of chars}
Lowercase letters count: {number of chars}
Uppercase letters count: {number of chars}
Other symbols count: {number of chars}

Зад.2
Иван е изобретил машина на времето и иска да я изпробва като пътува във времето до България в специфичен ден от годината, в периода от 1700 до 2700 година.
От 1700г. до 1915г., България официално използва Юлианския кландар, а от 1917г. - Грегорианския календар. Преминаването от Юлиянския към Грегориянския календар е било през 1916г., когато денят след 31 март е бил 14 април. Това означава, че през 1916г., 31 март е бил 91-я ден, а 14 април е бил 92-я ден от годината (ВАЖНО - 1916г е високосна).

И в двата календара, Февруари е единствения месец с променливи дни (29 дни по време на високосна година, и 28 през всички останали). В Юлианския календар високосни години са всички, които се делят на 4, а в Грегорианския високосни са всички, за които важи следното:
• Делят се на 400
• Делят се на 4, но не се делят на 100.

За година - x, и ден от годината - y, намерете датата на деня в посочената година според официалния календар на България през посочената година. Изкарайте датата на конзолата във формат dd.mm.yyyy, където dd е двуцифрен ден, mm е двуцифрен месец, yyyy е x.

Пример:

Ако x = 1996, y = 177, 1996 е високосна година (тъй като се дели на 4, но не се дели на 100 и е през времето на Григорианския календар) датата, която трябва да изкараме на конзолата е: 25.06.1996.

Входни параметри:
• year (x): integer
• day (y): integer

Ограничения:
• 1700 ≥ x ≥ 2700
• 1 ≥ y ≥ 365 (366 при високосна година).

Изходен формат:
Изкарайте пълната дата за посочения ден и година във формата - dd.mm.yyyy където dd е двуцифрен ден, mm е двуцифрен месец, yyyy е x.

Зад 3 ООП:
Калоян е собственик на хотел и има нужда от система за управление. Направате система с класове и йерархия, за да улесните работата на мениджъра.
Подберете най-добрите типове базирано на имената на полетата (Например за поле name, използвайте String, за number - int).

Имплементирайте всички класове в един клас Solution.

Структурата трябва да изглежда така:

```
abstract Room (
  roomNumber,
  pricePerNight
)

Suite : Room (
  constructor() {
      // When creating, initialize pricePerNight to 50
   }
)

Apartment : Room (
  constructor(roomNumber) {
  // When creating, check if roomNumber > 10 and if it is, set the pricePerNight = 150 (VIP
apartments). If not, set the pricePerNight to 100
   }
)

abstract HotelPersonnel (
  workerId,
  name,
  abstract greet method which takes a tourist and prints a greeting
)
```

Cleaner : HotelPersonnel (
  list of cleaned rooms,

  greet method which prints message: “Enjoy your stay,, {tourist name}”

  clean method which takes a room an adds it to the cleaned rooms list
)

Receptionist : HotelPersonnel (
  list of taken rooms,

  greet method which prints message: “Welcome to the hotel, {tourist name}”

  assignRoom method, which takes a Tourist and a Room object.
)

Tourist (
  name,
  room,
  stayDays,
  isAllInclusive

  constructor(name, stayDays, isAllInclusive)

  relax method, which checks if the tourist is staying with all inclusive. If they are, the method prints “Feels good to make fat stacks coding”, If they are not, just print a happy face :)
)

Зад 4 БОНУС:
Шерлок счита един низ за валиден, ако всички символи в него се появяват еднакъв на брой пъти. Също така е валиден, ако може да се премахне 1 символ от 1 място и останалите символи се появяват еднакъв на брой пъти. По даден низ s, определете дали е валиден. Ако е изпишете YES на конзолата, в противен счучай - NO.

Примери:
s = abc
Това е валиден низ, защото символите се появяват еднакъв на брой пъти = { a: 1, b:1, c:1 }.
s = abcc
Това е валиден низ, защото можем да премахнем един символ 'c' и остават по един символ от всички останали.
s = abccc
Този низ не е валиден, защото можем да премахнем само един символ 'c', което ще ни остави с { a: 1, b:1, c:2 }.

Входни параметри:
• s - String

Ограничения:
• 1 ≤ s ≤ 10000 (дължина на низа)
• Всеки символ s[i] ∈ ascii[a - z]

Изходен формат:
Ако низа е валиден - YES, в противен случай – NO

Kaufland Example Contest

1. Bar Star

Bar Star is a famous place where all the football players are hanging out. It servers two kinds of drinks: **Beer and Wine**. Every night the bar is **starting with 200 beers and 300 bottles of wine**. There are constant **orders and deliveries** through the night.

Your job is to calculate how many orders, deliveries and bottles are remaining at the end of the evening.

On a **new line** you will receive the drink and how the amount has changed. For example:

Input:

Beers: 25	It means a delivery of 25 new beers
Beers: -10	It means an order of 10 beers
Wines: 5	It means a delivery of 5 new wines

Basically, the orders are with **negative sign** and the **deliveries are positive** numbers.

When you receive a new line with the word **END**. You should terminate your program and print the remaining drinks, the number of orders and the number of deliveries. For example:

Input:

END	Terminate your program

Okay, here is an **example input data with the expected output**:

Beers: 30 Wines: -40 Wines: 15 Beers: -60 Beers: 10 END	Wines: 275 Deliveries: 1 Orders: 1 Beers: 180 Deliveries: 2 Orders: 1

2. Math Superstar

Cylinders! How can one not love them. Your task is simple. Calculate the **volume** of two cylinders and print the one with **bigger** volume in **centimeters**. If the volumes are **equal**

print any of them. Round the results **up to the second sign**. All the incoming data is in **millimeters**.

You might also need the **formula**:
$V = \pi * r^2 * h$

The input is a single line as follows: **r1,h1,r2,h2**. For example:

40,70,30,80	First, transform the data to centimeters. V1=351.86 V2=226.19. The first cylinder has a bigger volume.

So here is an example input and output.

35,67,22,109	257.85

3. Star Wars Design

In a galaxy far far away there are **3 types** of battleships.

- Outrider
- Millennium Falcon
- Ebon Hawk

Each ship has the following properties:

1. Id - number in the range [-9,223,372,036,854,775,808,+9,223,372,036,854,775,807]
2. Name - text
3. Color - Black or White
4. Attack damage - number in the range [-2 147 483 648,+2 147 483 647] 5.

Life - number in the range [-2 147 483 648,+2 147 483 647]

6. Shield - number in the range [-2 147 483 648,+2 147 483 647]
7. Capacity - number in the range [-2 147 483 648,+2 147 483 647]

Each ship has the following functions:

- Attack(battleship) - **attacks** a ship. The attacked ship should take damage and has a reduced **Shield** and/or **Life**.
- TakeDamage(damage) - the **damage** is taken first from the **Shield** and then from the **Life**.
- IsDestroyed() - returns true if the **Life** is less or equal than 0.

Each ship has the following characteristics.

- Outrider

    - Has **Plasma Defense** property - number in the range [-2 147 483 648,+2 147 483 647]
    - Receives **50 damage** less when attacked.
    - When it is created **receives 100 Life more**.
- Millennium Falcon
    - Has **Dodge** property - number in the range [-2 147 483 648,+2 147 483 647]
    - When attacks it **doubles** its damage.
    - When attacked **receives** 200 damage more.
- Ebon Hawk
    - Has an extra function **Heal(int life)** - increases its Life.

Your task it to **create** all the required classes and implement their methods following good **OOP** practices. Use appropriate **interfaces** and **abstractions**.

4. Star Mines

Mines are falling from the stars. That's all. Well almost. You are given a battlefield which is a **two dimensional array**. Each cell has **100 Life**. Mines are falling over the battlefield reducing the life of each cell with a given amount. All **neighbour cells** then have reduced **Life by 10**.

On the first row you will receive the dimensions of the array - rows and columns.

5, 5	It means an array with 5 row and 5 columns.

On the next rows you will receive the coordinates of the attacked cell.

3, 2, 50	The cell with row index 3 and column index 2 is attacked with 50 damage. All the cells around it(neighbours) are taking 10 damage.

This is how the **neighbours** of each cell are defined.

	N	N	N	
	N	Cell	N	
	N	N	N	

You **cannot** have cells with **negative Life**. If the attacked cell is **outside** of the array do **nothing**.
When you receive a line **GAME OVER** terminate your

program. Finally, **print** the array.

Here is an example **input**.

3,3	Create an array
2,1,30	Attack the cell at those coordinates. All neighbours take 10 damage.
0,1, 40	Attack the cell at those coordinates. All neighbours take 10 damage.
2, 5	Outside of the array. Do nothing.
GAME OVER	Print the array.

Here is how the **output** should look like.

90	60	90
80	80	80
90	70	90

Bonus task:

If the received damage is between **1 and 30** neighbours should take **5** damage. If it is between **30 and 70** neighbours should take **10** damage and finally, if it is over **70** the neighbours should take **20% of the damage**.

1. Pravets Course Planning

You are tasked to help planning the next Pravets Academy by keeping track of the lessons. On the first input line you will **receive the initial schedule of lessons and exercises** that are going to be part of the next academy, separated by **space(s) and arrow " -> ".** But before the academy starts, there are some changes to be made. Until you receive **"start academy"** you will be given **some commands to modify the course schedule**. The possible commands are:
`Add:{lessonTitle}` – add the lesson to the end of the schedule, **if it does not exist**.
`Insert:{lessonTitle}:{index}` – insert the lesson to the given index, **if it does not exist.**
`Remove:{lessonTitle}` – remove the lesson, **if it exists.**
`Swap:{lessonTitle}:{lessonTitle}` – change the place of the two lessons, **if they exist.**

Input / Constraints

- first line – the initial schedule lessons – strings, separated by **space(s) and arrow " -> "**
- until **"start academy"** you will receive commands in the format described above

Output

- Print the whole course schedule, each lesson on a new line with its number(index) in the schedule:
  `"{index}. {lessonTitle}". Initial index should start from 1.`

Examples

Input	Output
Spring Boot -> Angular -> SQL Add:GoLang Insert:HTML:0 Remove:SQL start academy	1. HTML 2. Spring Boot 3. Angular 4. GoLang
Input	**Output**
HTML -> Spring Boot -> Angular Swap:HTML:Angular Swap:Spring Boot:SQL Insert:HTML:0 start academy	1. Angular 2. SQL 3. HTML 4. Spring Boot

2.Pravets students

Write a program, which keeps information about **students** and **their grades**.
You will receive **n pair of rows**. First you will receive the **student's name, after that you will receive his grade. Check if student already exists, and if not, add him**. Keep track of all grades for each student.
When you finish reading data, keep students with **average grade higher or equal to 4.00**.
Order filtered students by **average grade in ascending**.
Print the students and their average grade in format:
`"{name} -> {averageGrade}"`

Format the average grade to the **2nd decimal place**.

Examples

Input	Output
5 5, 4, 5, Ivan 5.5, 3, 6, 5, Ivan 4.5, 3.5, Vladi 6, 5, 3,5 VLadi 3, 5 Emo	Emo -> 4.00 Vladi -> 4.50 Ivan -> 4.79

Input	Output
5 3.5, 4.5, Todor 3.5, 6, 5, Todor 4, 3.5, 6, 5, Plamen 5.5, 4.5, Tanya 5, Plamen	Todor -> 4.50 Plamen -> 4,70 Tanya -> 5.00

3.Pravets managing

Kaloyan is the Pravets school principle and needs some help with managing the school. Make a hierarchy of classes to make his job easier. Choose the best types based on the names of the fields (example: “name” should be String and “number” should be int)

Structure:

```
abstract Room (
  roomNumber,
  studentCapacity
)

ClassRoom: Room (
  constructor() {
     // When creating, initialize studentCapacity to 25
   }
)

PresentationRoom: Room (
  constructor(roomNumber) {
  // When creating, check if roomNumber > 10 and if it is, set the studentCapacity = 50. If not, set the studentCapacity to 75
   }
)

abstract Worker(
  workerId,
```

name,
abstract greet method which takes a student and prints a greeting
)

Principle: Worker (
number of students yelled at

greet method which prints message: “Don’t run in the hallways, {student name}”

yell method, which takes a Student object (student gets yelled at) and increases number of students yelled at
)

Janitor: Worker (
list of cleaned rooms,

greet method which prints message: “Don’t step on the wet floor, {student name}”

cleanRoom method, which takes a room and adds it to the clened rooms list
)

Teacher: Worker (
list of rooms where they teach,

greet method which prints message: “Hello, {student name}”

teach method which takes a Student (assigns this room and teacher) and a Room (adds it to rooms list)
)

Student (
name,
age,
Room,
Teacher,
attendsSports
yelledAt (default false)

constructor(name, age, attendsSports)

sport method, which checks if the student attends sport activities. If they are, the method prints “Feels good to be active!”, If they are not, just print a happy face 😊)

Kaufland Contest

Deadpool quiz

Deadpool wants you to test the mental abilities of the new heroes. He will come up with a **random number** between **1 and 100**. They have to guess that **number**. Your job is to write a program that does that for him.
Your program should generate a **random number** at **start time**. For every guess the program will either say "**high**" in case of a **higher number** or "**low**" in case of a **lower number** and then ask for another **input**.
At the **end** of the program the **number is revealed** along with the **number of guesses** it took to get the **correct number.**

Example:
Let's assume the **random number** is **60**.

Input	Output
34	low
67	high
60	Correct, attempts = 3

Ultron's trap

Ultron doesn't want you to help Deadpool. Moreover he wants you dead.

You woke up locked in a room that needs the correct password to escape. You are also given a riddle that looks exactly like an **array**. In order to retrieve the password you need to **reverse** its values. There are some catches:

- You **cannot swap** an **odd** number with an **odd** number
- When you swap an **even** with an **even** number you **multiply** the values by **2**
- If the **array length** is an **odd number** update the number in the middle (**last index/2**) to **0**

On the **first line** you will receive the numbers of the array.

Example:

Input	Output
5,2,3,4,5,6,8	8,12,3,0,5,4,5

Explanation:

Initial array:

5	2	3	4	5	6	8

Final array:

8 **(swap with 5)**	12 **(6*2)**	3 **(no swap)**	0 **(the array's length is 7(odd), so the element at position (6/2) is 0**	5 **(no swap)**	4 **(2*2)**	5 **(swap with 8)**

Iron Man Laboratory

Since you've earned your freedom it is time... to go back to work. Tony Stark wants you to build a new battle **equipment** from him.

Every **battle equipment** has the following **properties**:

- **Life** - a number between 0 and $+2^{31}-1$
- **Defence** - a number between 0 and 100
- **Attack Power** - a number between -2^{31} and $+2^{31}-1$
- **Name -** text
- **Flying Mode** - Stealth, Offensive, Defensive
- **Battery Life** - percentage between 0 and 100

And the following actions:

- **Attack** - returns the attack power
- **Dodge** - returns the defence multiplied by 0.08
- **Recharge** - increase the battery life with 20.35%

Moreover, we have **3 types** of equipment.

War Model:

- Has doubled **attack power** when created
- **Life** is reduced by 10% when created
- When **attacking** it adds 10 damage

Schnell Model:

- Has **Speed** property in the range -2^{31} and $+2^{31}-1$
- **Dodge** is multiplied by **0.092** instead of 0.08

- When **attacking** it does 5 damage less

Peaceful Model:

- **Life** is doubled when created
- Cannot **dodge**

Create the required classes by using good **OOP** practices. Make the necessary **validations**.

Saving of Lady Marvel

You got away from the Ultron's trap but the same thing cannot be said for Lady Marvel. She is imprisoned by Dr. Doom servants. Decrypt her messages and send them to the Avengers for help.

On the **first line** you will receive the **total number** of messages. On the next **couple of lines** you will receive her messages. Some of them are saying the **truth** and others are **false**.
A true **message** has the following pattern:
{exactly 5 numbers}{?}{true message}{!}{exactly 3 letters}

Your job is to **print** all the true messages on a new line.

Example:

Input	
5	
34141?Send some help!dms	The message is correct
3414??I am home	The message is wrong
66666?I am in the doom tower!ggg	The message is correct
Solve my riddle	The message is wrong
54256?Dont rush!xxxs	The message is wrong

Output
Send some help I am in the doom tower

Tic-Tac-Toe Game with Mystique

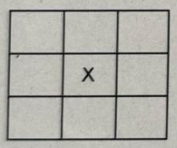
Tired of all the little hero's games? So is Mystique. She can be pretty hot so you decide to impress her with… a game.

Create a **Tic-Tac-Toe** game.
The game is played on a grid (**2d matrix**) that is **3 by 3** squares.
You are the 'X' player and your Mystique is the 'O' player. Players take turns putting their marks **('X' or 'O')** in the **empty matrix squares**.
The first player which **gets 3 of the marks in a row** (**up, down, across, or diagonally**) is the winner.
When all **9 squares are full**, the game is **over**. If **no player has 3 marks in a row**, the **game ends** in a tie.
If one of the players **'X' or 'O' has a row** then you should print: **Player (either 'X' or 'O') won the game**!

Example:

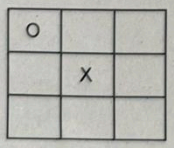
Turn 1

Input
Player 'X', enter your move: **2 2**



Turn 2

Input
Player 'O', enter your move: **1 1**



Turn 3

Input

Player 'X', enter your move: **1 3**

O		X
	X	

Turn 4

Input

Player 'O', enter your move: **3 1**

O		X
	X	
O		

Turn 5

Input

Player 'X', enter your move: **2 2**

Output

This move at (2,2) is not valid. Try again...

Input

Player 'X', enter your move: **2 3**

O		X
	X	X
O		

Turn 6

Input

Player 'O', enter your move: **2 1**

O		X
O	X	
O		

Output

Player 'O' won the game!

Good luck! She is **worthy**! So are our **traineeship positions**.