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**.