THE CRC ROBOTICS COMPETITION

PROGRAMMING CHALLENGES

INVICTA 2021

A program of

AEST EAST

ALLIANCE POUR L'ENSEIGNEMENT DE LA SCIENCE ET DE LA TECHNOLOGIE
EDUCATIONAL ALLIANCE FOR SCIENCE AND TECHNOLOGY

Table of Contents

CONTEXT

The year is 2087. Following the international community’s failure to reduce the production of greenhouse gases, the Earth has passed a point of no return and is becoming more and more uninhabitable with each passing minute. Tides are rising, continents are shrinking and pandemics are multiplying. However, we still have one hope: you. You have discovered a potentially habitable exoplanet in a nearby planetary system.

On your way to this new planet that could save us all, you hit a massive asteroid, which caused you to crash into the nearest planet, costing you your only engineer-mechanic. The planet is composed mostly of a strange rock that is unknown to date. So, while your expedition robot collects samples of this rock that could advance science, you will have to repair the different parts of the ship that have been damaged.

Your task will be to repair these parts of the ship as quickly as possible in order to leave as soon as possible. If you do everything right, you should be able to arrive in February of next year. Hurry up, astronauts, you can still save the human race!

Problem #1 - The Entire Division (1 point)

Sometimes when we divide integers, we get a decimal answer, and in some cases we may not be interested in that answer. For example, we want to separate a basket of oranges evenly into several smaller baskets, but we don't want to separate the oranges themselves. So, it is possible that we are not only interested in the whole result, but also in what is left at the end (or what we can no longer separate).

Instead of just doing the division, we will have to subtract the divisor from the dividend until we can no longer do so. The number of subtractions made becomes the result of the division and the number that remains after we can no longer subtract becomes the remainder. It is enough to visualize that the program performs what we would normally do in a bracketed division. This will give us the whole result and the remainder of the division, which is itself very useful in programming.

Input specification

The first line is the number of tests to perform. The following lines each contain two numbers, separated by a space. The first is the dividend, or the number to be divided, and the second is the divisor, or the number by which the other is divided.

Output specification

For each division, the answer should be given in the form "a remainder b" where a is the result and b is the remainder. If there is no remainder (b = 0), just the result is given.

Example of input

```
5
56 18
34 7
89 128
581397 869
1159029 9423
```

Example of output

```
3 remainder 2
4 remainder 6
0 remainder 89
669 remainder 36
123
```

Explanation of the first output

We subtract 18 from 56 once, leaving us 38. We can then subtract it a second time, leaving us with 20. We do the same thing again to get 2. Now 18 cannot be subtracted, so 2 becomes the remainder. 18 has been subtracted 3 times in the process, which gives 3 remainder 2. We can indeed check with 18x3 + 2 = 56.

Problem #2 - The Ship's Model (2 points)

Your task here will be to create the program that allows you to view a miniature version of your ship on a screen in the cockpit. This small version of your ship is mainly used to quickly and easily see if there is a problem elsewhere in the ship by identifying the area and the problem on the model.

Fortunately, you will only need to create the "skeleton" and the data that has been saved from the program will take care of the rest.

Thus, the "skeleton" is a single closed shape consisting of a rectangle with a triangle on top or, if you imagine it in three dimensions, a cylinder with a cone on top.

Input specification

The first line indicates the number of tests performed to validate your program. All subsequent lines contain two numbers between 1 and 10 separated by a space. These numbers represent the height of the rectangle and the height of the triangle, respectively.

Note: It is important to note that, because the output in the console of a problem is usually monospace (all letters and symbols take up exactly the same space), the height of the triangle is therefore also half its width and therefore half the width of the rectangle (minus the sides).

Output specification

The output will be, for each set of dimensions (each input line), a "drawing" formed by "_" and "|" for the rectangle, and "/" and "\" for the triangle.

Example of input

```
2
5 3
7 5
```

Example of output

```
   /\
  /  \
 /    \
|      |
|      |
|      |
|      |
|______|
    /\
   /  \
  /    \
 /      \
/        \
|        |
|        |
|        |
|        |
|        |
|        |
|________|
```

Explanations of the first output

5 is specified for the height of the rectangle, which makes the sides of the rectangle contain 5 "|". Then, 3 is specified for the height of the triangle, which makes the width 6 excluding the walls (it takes 3 characters to reach the height and thus 3 to go down). So we have 3 "/" and 3 "\" that make up each side of the triangle. Then, the width of the rectangle at the bottom must match, so there are 6 "_".

Problem #3 - Simulate Switches (3 points)

In any situation, many programs need to simulate the operation of a switch to activate or deactivate certain functions. Such a switch, if it were physical, would have two positions for the lever or button that activates it: a position that means the switch is on and a position that means the switch is off.

Now, a very efficient way to simulate this in programming is to use a Boolean (a variable that can be True or False). So when you "move" the lever, if it was False, it becomes True, and if it was True, it becomes False.

So, you will need to simulate 3 of these switches simultaneously, so that any of the 3 can be activated or deactivated in any order.

Input specification

The first line is the number of tests to be performed. Each subsequent line corresponds to a list of commands to be performed, all separated by spaces. The commands are S1, S2 and S3, and correspond to their respective lever. **By default, the switches are disabled at the beginning of each test.**

Output specification

For each set of commands, you will need to send the result of each switch after all commands have been performed. The result will be reported as True or False and the switches must be in order.

Example of input

```
3
S1 S3
S2 S1 S2 S3 S3
S3 S1 S1 S2 S3 S2 S3 S1 S2 S1
```

Example of output

```
True False True
True False False
False True True
```

Explanation of the first output

All the switches are set to False at the beginning, then we activate the first and the third (only once, so we don't turn it back off), finally leaving us with True False True.

Problem #4 - Measuring the Temperature (4 points)

To operate the vessel efficiently, it is necessary to measure the temperature. However, the company that created the thermometer is American and provides the temperature in degrees fahrenheit. The on-board computer must make the conversion to Kelvin, a much more scientific unit.

To convert a measurement in Fahrenheit to Celsius, the formula is: $C = 5(F - 32) / 9$
To convert a measurement in Celsius to Kelvin, the formula is: $K = C + 273.15$

Your program must take a measurement in Fahrenheit and convert it to Kelvin.

Input specification
The first line contains an integer 1 <= N <= 100,000 which indicates the number of measurements taken by the sensor. The next N lines contain a real P, which represents the measurement in Fahrenheit.

Output specification
For each line, the output is a real number with a precision of one decimal place corresponding to the temperature in Kelvin.

Example of input

```
3
1.0
-3.7
100.0
```

Example of output

```
255.9
253.3
310.9
```

Explanation of the first output
By the first equation, 1.0^{o}F = -17.2^{o}C and, using the second equation, -17.2^{o}C = 255.8^{o}K.

Problem #5 - Livable Temperature (5 points)

To find a livable planet, the exoplanet detector takes the temperatures of planets in the universe at different locations and moments to get six different measurements. To ensure that the temperatures on this planet are accurate, your program will have to give the average temperature to 1 decimal place of accuracy. If the temperature is not suitable for a human, the message "danger" must be returned. For the temperature of the planet to be valid, the average must be between 10 and 30 (inclusive) degrees Celsius, and the greatest difference between two measured temperatures must not exceed 60 degrees.

We will need to find the average temperature and ensure that it is acceptable for a human.

Input specification

The first line specifies an integer n between 1 and 10,000, corresponding to the number of tests performed. The next n lines contain 6 real numbers, which are the temperature measurements on the planet.

Output specification

The expected output is a real number rounded to 1 decimal place if the mean and the largest deviation respect the norms. Otherwise, the expected value is "danger".

Example of input

```
3
4.3 2.3 51.3 34.5 29.0 12.4
-12.5 2.9 17.6 -16.3 7.7 20.5
-24.0 42.4 26.5 29.9 17.5 17.6
```

Example of output

```
22.3
danger
danger
```

Explanation of the first output

22.3 is the average temperature, because it is within the acceptable standards for the average temperature, and the maximum temperature deviation is 49 degrees Celsius, which is also acceptable. Both conditions are met, so we provide the average temperature.

Problem #6 - The Barcodes (6 points)

In the ship, all artifacts and samples found during the mission must be classified with a barcode that can be scanned to access the information quickly in the database. A specific number is assigned to each sample once collected. This number is given according to our classic decimal system. You will first need to convert it to a 16-bit binary number, and then change the 0's and 1's to spaces and bars, respectively.

Input specification

The first line is an integer representing the number of tests to perform. Each subsequent line corresponds to a test and consists of a positive integer n between 1 and 65535.

Output specification

Your code should give the barcode corresponding to the number of each test. The barcodes will be surrounded by #'s to clarify the beginning and end of the label.

Example of input

```
3
17238
9819
39385
```

Example of output

```
# |    || | | || #
#  |  ||  | || ||#
#|  ||  ||| ||  |#
```

Explanation of the first output

17238 gives 01000011010110 in 16-bit binary and, by replacing the 1's with bars and the 0's with spaces, we obtain the first line above.

Problem #7 - Balance the Thrusters (7 points)

The propulsion system of the ship is composed of two perpendicular engines. The main engine A handles the forward thrust while engine B handles the lateral thrust. The on-board computer system is a prototype which was not updated with the system of the rocket. Therefore, it gives the navigation data in "tank" control, as if the 2 rockets were propelling in the same direction.

You have to convert the navigation commands from the (Left, Right) format to the (Forward, Lateral) format. By convention, the positive direction of the lateral engine will turn the ship to the right. So we want to convert commands from the format **(a, a+b)**, where b simply symbolizes a difference in thrust between the two engines, to the format **(2*[weaker engine], b)**. If you look more closely, you can see that equal thrusts move the ship forward while the excess on one side turns the ship, which is converted to a lateral coordinate.

Input specification

The first line contains an integer 1 <= N <= 10,000 which indicates the number of tests to be performed. The next N lines contain a real 'G', the left thrust force, and a real 'D', the right thrust force. The input precision is to the nearest tenth.

Output specification

For each input, the expected output is a real 'A' which describes the forward thrust and a real 'L' which describes the lateral thrust. The expected accuracy is also to the nearest tenth.

Example of input

```
5
12.2 13.4
23.2 0.0
3.7 3.7
-1.0 -1.2
2.1 2.2
```

Example of output

```
24.4 1.2
0.0 -23.2
7.4 0.0
-2.0 -0.2
4.2 0.1
```

Explanation of the first output

We have 1.2 times more power in the right engine, which gives us +1.2 power for the lateral engine. Then, the remaining 12.2 on each side balances out in direction to move the ship forward by 24.4.

Problem #8 - Distance Calculation (8 points)

Living in a three-dimensional world, we need three coordinates to describe a single position, no matter what coordinate system we use. There are more complicated coordinate systems that are better suited for space and astronomy, but we will use Cartesian coordinates here as the simplest and best known system.

Thus, any point in any three-dimensional space can be expressed in (x,y,z) coordinates. However, the ship will not travel the path to a coordinate point by traversing the axes one by one because it would be much slower. Thus, we want to calculate the distance that the ship must travel in a straight line between two positions.

Knowing that the x, y and z axes are perpendicular, doing a little trigonometry, we can see that the total distance can be calculated as follows:

$$d = \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2}$$

where Δx is the distance between two coordinates on the x-axis (x_1 - x_2). It's the same thing for y and z.

Finally, depending on whether we are very far or extremely close to our target, we would potentially like to have several different distance units at our disposal.

The units used will therefore be the kilometer (km) for small distances, the astronomical unit (AU), the distance from the Earth to the Sun for medium distances, and the light year (AL), the distance that light travels in one year, for longer distances. There are even larger units, but we won't deal with them here.

The conversion rates you will use are as follows:

$$1\,UA \approx 150\,000\,000\,km$$
$$1\,AL \approx 9\,461\,000\,000\,000\,km$$
$$1\,AL \approx 63\,073\,UA$$

Input specification

The first line contains the number of tests to be performed. The following lines contain three main data. The first data item will be the unit of measurement in which the coordinates will be given. Then, the second data will be the position of the vessel in the Cartesian coordinate system. Finally, the last data will be the position of the point where we want to go and with which we have to calculate the distance.

Output specification

In the output, we will require the distance between the two points expressed in the most appropriate unit and rounded to the nearest thousandth. Differences of ±0.003 units are acceptable.

Distance *d*	Required Unit
$d < 0.1$ UA	Kilometers (km)
0.1 UA $\leq d <$ 0.1 AL	Astronomical Unit (UA)
0.1 AL $\leq d$	Light-Year (AL)

Example of input

```
5
km (27486 53294 2435) (57293 189070 289065)
UA (-0.37 1.27 8.92) (0.77 -2.82 5.61)
km (-302 9047 -65) (-257988303 287302 807944832)
AL (2.5608 3.8729 -0.0192) (2.5712 3.8477 -0.0185)
AL (234 -845 99) (58 -1133 -202)
```

Example of output

```
318559.785 km
5.384 UA
5.654 UA
1720.044 UA
452.240 AL
```

Explanation of the first output

We put it all in the distance formula, which gives us 318559.785km. Since 0.1 AU corresponds to about 1,500,000 km, we are below and remain in km for the output value.

Problem #9 - Controller Selection (9 points)

To activate your ship's highest speed, you need to reach full engine power by activating all the levers in the engine room. Since the controls are arranged in a circle, you activate them starting from the second lever, as the first is too difficult to activate. You then activate them in ascending order of the controllers. However, when you activate one controller, you cannot activate the next one immediately, because you will receive an electric shock. So, you must always avoid the controller next to the one you just activated, even if the numbers to mark the controllers are not consecutive. As the room is circular, after the first round, you continue until you get back to the first numbers and start again from the beginning in a circle following the same logic of always skipping a lever. The count of the levers goes from 1 up to and including the given value. However, you start by activating the second lever first.

So, you are looking for the last controller to be activated depending on the number of controllers in the engine room.

Input specification

The first line indicates the number of test cases to be run. The following lines will contain an integer representing the number of controllers in the engine room.

Output specification

The output will be an integer corresponding to the last controller to be activated without getting electrocuted (following the rule of one controller out of two).

Example of input

```
3
13
9
8
```

Example of output

```
11
3
1
```

Explanation of the first output

First, we go around the table, removing all even numbers, leaving the controllers [1, 3, 5, 7, 9, 11, 13]. After passing on 12, we dodge 13 and lower the first lever. In the same way, we lower 5, 9, and 13. Now, we have to start a third round from the beginning with levers [3, 7, 11]. We skip 3 to lower 7. Finally, we skip 11 and lower 3, leaving us with 11 as the last lever to activate.

Problem #10 - Start the Reactor (10 points)

You are aboard your spaceship for a very long journey, which requires your ship to have its own power source. The ship has a rather large reactor (of unknown type, but you don't care) which is capable of providing a lot of energy.

However, this reactor unfortunately shut down during the crash. Because it is potentially dangerous, there is a 20-digit security code to restart it. You can also see that there are two digital displays on the engine: one on the left that can display up to 4 digits and another on the right that can display only 2 digits.

Having witnessed much of the making of this reactor, you know that it is the Syracuse suite. The indicator on the left shows the number of the Syracuse sequence, and the one on the right shows where to start the sequence in the security code. It is important to design a program which can calculate everything, because the two indicators, and thus the security code, change every 20 seconds, which leaves very little time for a human to calculate everything.

The Syracuse sequence is therefore an old mathematical sequence that takes any integer and always ends up reducing it to 1. We therefore proceed as follows for each instance (number) of the sequence:

- If the integer is even, we divide it by 2
- If the integer is odd, we multiply it by 3 and we add 1 to it

It will thus be necessary to make a program which calculates the Syracuse sequence for a certain number n and displays it (without spaces between each instance) starting from the number displayed on the right (we will take the number n as being the instance #1). It is also important to note that the Syracuse sequence does not end when the number 1 is reached but continues until infinity with the period 1 4 2.

Input specification

The first line corresponds to the number of tests performed. The following lines correspond to the two parameters displayed on the indicators, in the order in which they were stated above.

Output specification

Your program should output a series of twenty digits that will serve as the secret code to activate the reactor.

<u>Example of input</u>

```
4
0014 05
2587 22
8191 85
9999 37
```

<u>Exemple de sortie</u>

```
34175226134020105168
20741037311215567783
20230101153034615173
31995847914387192158
```

<u>Explanation of the first output</u>

We ask for the Syracuse sequence of the number 14, which goes as follows:
14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, 2 ...
We want to build a security code of 20 characters from the 5th digit of the sequence, that is 34. So, we have the code 34175226134020105168 where we stop, because we have reached the 20th digit of the code.

Problem #11 - Consumption Control (11 points)

Now you have to make sure that your ship does not consume too much electricity. To do this, you will be given a table containing the number of active hours per day of each room. You will also be given a table containing the hourly consumption of each room.

Thus, you will first have to determine the total consumption of each room during the given period, and then you will have to determine the total consumption of the whole ship. The hourly consumption is in Watts (W) which, along with the number of hours, should give you watt-hours (Wh).

Input Specification

The first line contains the dimensions of the active hours table (max 50 x 50), so the first number is the number of rows (i.e. the number of days) and the second number is the number of columns (i.e. the number of rooms). This table is given to you directly below. Finally, the second table, the hourly consumption table for each room, is given to you after the first table. It is important to note that the second table has as many columns as the first one or this challenge would not make sense.

Output specification

You will need to provide a table containing the total consumption of each room for the given period. Then, below this, you will need to provide a number symbolizing the total consumption of the ship in Wh (so the sum of the table your code just gave). All your outputs must be rounded to the nearest Wh. Differences of a few Wh are acceptable.

Example of input

```
14 17
4.23 22.4 11.2 13.0 19.0 7.25 20.8 11.1 4.99 5.55 10.9 8.09 16.7 3.10 11.4 19.9 5.41
2.14 1.17 15.4 13.0 8.92 2.20 17.8 20.2 17.9 7.88 4.02 20.0 4.21 5.98 1.52 1.85 18.8
13.7 2.92 0.96 21.5 13.4 19.3 23.2 11.2 18.1 21.7 21.5 22.9 20.5 20.0 12.2 23.3 16.5
4.59 13.2 1.46 18.9 20.7 5.68 7.85 22.5 16.9 2.84 16.1 19.0 21.6 0.61 18.1 0.88 11.7
14.9 23.7 6.51 0.70 23.1 3.57 21.6 11.4 2.35 20.8 12.3 2.37 6.29 13.3 3.15 22.0 22.3
8.03 3.77 7.81 11.3 23.1 8.57 20.9 14.8 19.6 6.08 18.3 0.29 12.3 10.8 22.6 22.5 11.0
1.64 18.1 0.43 18.8 5.86 10.2 19.1 21.3 1.82 14.1 10.7 4.31 13.8 18.9 18.5 17.4 18.4
18.0 1.75 15.7 18.9 3.55 20.3 9.66 16.8 15.0 3.86 8.76 19.1 17.5 4.77 11.1 9.67 17.2
20.0 4.46 16.7 10.2 6.09 18.5 14.6 14.7 9.14 16.8 21.6 9.99 20.5 19.5 7.78 16.5 5.58
9.44 7.63 14.9 17.8 1.86 18.9 13.9 12.3 11.5 5.48 17.6 2.77 8.03 24.0 12.9 23.8 8.43
5.70 2.29 8.76 14.6 18.2 3.74 7.43 15.9 7.65 16.1 5.46 20.9 17.3 13.1 5.00 0.33 4.06
2.34 16.9 17.7 12.8 8.25 12.1 9.15 3.92 21.5 0.33 11.8 1.49 16.4 2.04 12.2 6.20 0.45
19.1 15.3 14.8 4.46 18.4 0.04 23.7 6.59 22.3 18.9 12.0 4.18 9.47 14.9 18.9 19.0 22.4
```

```
5.52 6.96 3.15 0.89 6.59 3.07 15.7 1.50 6.17 10.7 19.8 7.26 15.2 19.2 8.32 1.93 11.0
10 25 60 60 100 100 100 100 100 350 550 850 1200 2000 2000 2300 3000
```

Example of output

```
1293 3514 8129 10611 17702 13342 22539 18421 17492 52892 104962 121252 239760 340400
327340 426098 519690
2245437 Wh
```

Explanation of the output

We add up all the hours of consumption of the first room (i.e. the first column), multiply the total by its hourly consumption and round up to the nearest Wh, which gives 1293 Wh. We do the same for all the columns, then add them together to get the total consumption of the vessel in Wh, which is 2245437 when rounded.

Problem #12 - Watering of the Plants (12 points)

To feed the crew, a greenhouse is present on the ship. Each plant must be watered according to a quantity determined by the on-board computer. However, only two pots are available and the quantity of water to pour is not one of the quantities of these two pots. It is also impossible to measure a partial quantity (impossible to add 1 liter of water in a 2-liter pot).

At each step, you can perform the following commands:
- Fill bucket 1 (FILL1)
- Fill bucket 2 (FILL2)
- Transfer the contents of bucket 1 to bucket 2 (POUR1)
- Pour contents of bucket 2 into bucket 1 (POUR2)
- Empty bucket 1 (DISCARD1)
- Empty bucket 2 (DISCARD2)

Input specification
The first line specifies an integer 1 <= N <= 10,000, which represents the number of tests described. The next N lines contain an integer 'Q1', which is the size in liters of the first bucket, 'Q2', which is the size in liters of the second bucket, and an integer 'T', which is the desired amount of water in one of the containers. The data is such that 1 <= 'Q1' < 'Q2' and 2 <= 'T' <= 'Q2'.

Output specification
Each line corresponds to the optimal sequence of operations to water the plant. For each operation, specify the action according to the name specified in the problem statement. At the end of the solution, one of the two pots must contain the required amount of water.

Example of input
```
3
3 5 4
7 23 5
39 80 2
```

Example of output
```
FILL2 POUR2 DISCARD1 POUR2 FILL2 POUR2
FILL1 POUR1 FILL1 POUR1 FILL1 POUR1 FILL1 POUR1 DISCARD2
FILL2 POUR2 DISCARD1 POUR2
```

Explanation of the first output

The 2 buckets have initially 0L and 0L, respectively. We fill the second bucket, which gives 0L and 5L. We empty the second bucket in the first one, which gives 3L and 2L. We empty the first bucket, to have 0L and 2L. Empty the second bucket into the first one, which gives 2L and 0L. We fill the second bucket, which gives 2L and 5L. The second bucket is poured into the first, giving 3L and 4L. The second bucket contains 4L, so the solution is valid.

Problem #13 - The Encyclopedia (13 points)

During your stay on the planet, you have a robot that will be at your side to collect samples of material on the planet. With the help of an internal encyclopedia, you will try to identify what materials certain samples are made of and in what proportions.

You will be given an encyclopedia containing important information on several materials and you will have to use the information on your sample to determine its composition. **Each sample will be composed of only two different materials.**

Input specification

The first line contains an integer n which corresponds to the number of entries (and therefore materials) in the dictionary. The following lines each contain a dictionary entry of the form [Name - Volume - Mass for the specified volume - Color], which are the properties of the materials. Two different materials will not have the same color.

The line after the dictionary corresponds to the number m of tests to be performed (or the number of samples to be identified). The following lines each contain a test of the form [Colors - Volume - Mass], that is, the properties of the sample.

The masses are in g and the volumes are in cm3 in the problem.

Output specification

For each sample, we expect a percentage rounded to the nearest hundredth of each of the two materials present of the form [X% material1 and Y% material2]. The composition is represented volumetrically (i.e. a sample is made of 5% of a material if that material takes up 5% of the sample volume). Differences of a few hundredths of a percent are acceptable. **The material most present in the sample is given first (highest percentage).**

Example of input

```
4
Oxyde de Fer - 300 - 1572 - Rouge
Quartz - 171 - 453.32 - Rose
Or - 89 - 1717.7 - Jaune
Argent - 237 - 2486.13 - Gris
3
Rouge/Jaune - 2689.22 - 17267.99
Rose/Gris - 394.55 - 3677.58
Jaune/Gris - 578.24 - 8894.87
```

Example of output

```
91.60% Oxyde de Fer et 8.40% Or
85.09% Argent et 14.91% Quartz
55.54% Or et 44.46% Argent
```

Explanation of the first output

As we find yellow and red in the sample, we can deduce that it is composed of iron oxide and gold. We can then find the proportions using the volume and mass of the sample, and the information in the internal dictionary. The only thing left to do is round the result and put the iron oxide in front, because it is more represented in the sample than gold.

Problem #14 - Encoding a Message (14 points)

Since electromagnetic waves such as light are transmitted through space as well as through the air, it would be possible for you to send messages (on radio waves, for example) to the Earth.

Now imagine that there is no planet between you and your recipient (the Earth) and thus nothing can block the passage of the waves. You can therefore transmit messages to the Earth and your ship is capable of doing so. The problem is that you would have to encode the messages to prevent them from being understood by everyone who perceives them.

In this problem, you will repair the message encoder of the ship. It will also be necessary that the key with which you encode your message is understood by its receiver. You will thus use, as agreed at the takeoff of the ship, the dates of sending and reception of the message (which will be calculated for you) to modify the value of the letters. Here is how you will calculate the new letter:

$$\#L = (D^*_{send}[i] \times \#L + D_{reception}[i]) \bmod 26$$

where $\#L$ is the number associated with the letter (A=>1, B=>2 … Z=>26),
$D^*_{send}[i]$ is the i-th digit of the corrected sending date (see below),
$D_{reception}[i]$ is the i-th digit of the reception date (see below)
and mod 26 marks the remainder of an integer division by 26 (because if the result exceeds 26, we are no longer in the alphabet).

For dates, the format used is YYYY/MM/DD. Thus, when we go through the letters of a message (we don't count spaces or special characters like the period(.)), we use the first digit of the year of the two dates to encode the first letter, the second digit of the date to encode the second and so on until the ninth letter, where we start the cycle again with the first digit of the year. (One will thus consider that $1000 \leq year \leq 9999$)

However, there is a major problem with this method. If the number used for the date of sending is **not** a prime factor of 26, at the time of decoding, the new letter can lead to several answers. **Thus, the number used for the date of sending must always be a prime factor of 26 and can therefore only be 1, 3, 5, 7 or 9. If some of the numbers given in the date of sending are incompatible, they will have to be corrected to the higher integer** (e.g. a 2 in the sending date gives 3).

Input specification

The first line will provide you with two dates in YYYY/MM/DD format. You will have to ignore spaces. The date on the left will be the sending date and the one on the right will be the date of reception of the message on Earth (yes, calculated in advance). Then, from the second line, you will have the message that you will have to encode. **You will have to keep all the special characters for the message while ignoring them for the encoding. The message will be given to you without accents on the letters to simplify. Finally, the only characters you will need to ignore and return in the encoded message are [, .] and space.**

Output specification

You will need to provide the encoded message following the exact same structure as the message you received while having changed all the letters according to the encoding formula. The capital letters must be kept.

Example of input

3185/11/25 3188/05/18

Vaisseau Invicta a la base terrestre. Nous avons trouve une exoplanete potentiellement habitable assez proche de notre location. Cependant, nous avons ete frappes par un asteroide, nous forcant a nous ecraser sur une planete environnante. Il semble aussi qu,il puisse y avoir presence de vie sur la planete sur laquelle nous nous trouvons. Je, attitre capitaine du vaisseau Invicta, m,occupe donc de reparer le vaisseau pendant que l,equipe de chercheurs recolte des echantillons pour tenter de confirmer ou d,infirmer cette possibilite. Informations supplementaires a ce sujet viendront sous peu.

Example of output

Qbkysjdi Doxacyd m mb zmsj igesaytwp. Zvvw mvtqy ksmivj lzr fpepqdzrua joypzkjapljngsu bmbnimima msxph ysmwhj mg spfte qtwfuken. Hpjrormny, qent qnosf gkf jtauwgh qqt us dykfneiip, zvvw lowjmsu q zozf glsqyew fie vdg pqdzrua gnabtvodmnyp. Am taubqp mntwa qz,bp yvkysj x mqpkt pwpyroig dj oar tot lf wpfoade xlt mbeieqkg spoy ntly ksmivtqy. Gf, qdtnitr dqjiydasf ri vfbyhfqi Isoaluq, u,ohjiyf renh mg efvmrjc pr wqasxpmn qazdfqd bva p,evlayf rg cmptliairx cglplde ipy rdbmnybpmpdy ptlt kfddew mg lpdliwnge po b,issaenat cjidr qmysngamjfg. Isseenqditqy hvvjljngsuqarjf m lf wijji ndfdbrtqd hpoy pjl.

Explanation of the beginning of the output

First we need to correct the send date from 3185/11/25 to 3195/11/35, which gives us the key [3, 1, 9, 5, 1, 1, 3, 5]. The date of reception gives us [3, 1, 8, 8, 0, 5, 1, 8]. We then start to progress in the message. 3x22 (V) + 3 = 69. 69 mod 26 = 17, which corresponds to Q. We then go through the message with the key. 1x1 (a) + 1 = 2, which corresponds to b (already in mod26). After the first word, we skip the space and arrive at the beginning of the second word. This is the 9th letter of the message, but our key has only 8 digits, so we start the key again at the beginning. Thus, 3x9 (I) + 3 = 30. 30 mod 26 = 4, which corresponds to D. We continue in this way, skipping anything that is not a letter and not counting these special characters in our encoding.

Problem #15 - Complex Number Calculator (15 points)

Having just about finished repairing the whole ship, you decide to take a break by exploring complex numbers. During your "break", you will have to code a program that will recognize complex numbers and basic operators ($+ - \times \div$). Don't panic! You won't need to code exponents, roots and priority operations. Your break isn't that long...

As for complex numbers, they usually take the form $a + bi$, but can also be written as a if $b = 0$. For example, *a* and *b* are real coefficients (don't forget that they can be negative) and *i* is the imaginary number $\sqrt{-1}$ (which doesn't exist in the real number space). So, we say that *a* is the real part of a complex number denoted *z*, and that *bi* is the imaginary part. The main thing to know about complex numbers is that $i \times i = -1$. Knowing this, we can start the elementary operations...

For addition and subtraction, it's simple: we add or subtract the real parts of the two complex numbers and it's the same for the imaginary parts.

$$z_1 \pm z_2$$
$$= (a + bi) \pm (c + di)$$
$$= (a \pm c) + (b \pm d)i$$

where *a±c* is the real part of the complex number *z* and *(b±d)i* is the imaginary part.

For the multiplication, **which will be noted by * in the operations**, it is necessary to make all the possible multiplications, as if we had $(a + bx) * (c + dx)$.

$$z_1 * z_2$$
$$= (a + bi) * (c + di)$$
$$= ac + adi + bci + bdi^2$$

The rest of the development is up to you, but what makes the multiplication so simple is that i^2 becomes -1, leaving *bd* real (but modified by a -). So, we complete the multiplication with, once again, a complex number of the form $a + bi$.

For the most **complex** operation, division (**which will be noted by / in the operations**), we must first introduce the notion of complex conjugates. For any complex number z, there is a complex conjugate z such that, if $z = a + bi$, then $\underline{z} = a - bi$. The advantage of multiplying by the complex conjugate is that the resulting number is real because the term in *i* cancels and the term in i^2 becomes real. So, when we divide one complex number by another, we will have to multiply the divisor and the dividend (so that it doesn't affect the result) by the complex conjugate of the divisor, leaving us with a complex number on top and a real number on the bottom. The real number is then distributed on each of the two parts of the complex number to reduce its coefficient. So we proceed as follows:

$$\frac{z_1}{z_2}$$
$$= \frac{z_1 * \underline{z_2}}{z_2 * \underline{z_2}}$$
$$= \frac{(a\ +\ bi) * (c\ -\ di)}{(c\ +\ di) * (c\ -\ di)}$$
where we obtain $c^2 - d^2 i^2$at the bottom. The denominator of the fraction thus becomes real. The rest of the development in the denominator and the development in the numerator is left for your personal enjoyment.

Input specification
The first line corresponds to the number of tests to perform. The following lines each correspond to a sequence of specific operations to be performed from left to right (operators will be used so that there is no issue in the priority of operations). There will be a space between all the complex numbers and the operators, and there will be no parentheses surrounding the complex numbers for simplicity. ($(a + bi)$ will be noted by $a + bi$)

Output specification
You will need to output the resulting complex number from the chain of operations as $a + bi$. Be careful! If $b = 0$, you should display only a and vice versa if $a = 0$ (bi). If both are 0, your program must display 0 and not nothing. If the coefficients *a* and/or *b* are not integers, they will be rounded to the nearest hundredth (i.e. numbers with commas).

Example of input
```
4
5+7i + 2-3i - -8+19i
6-2i * -7-4i * 9-13i - 1+6i
-5-8i / 1-i
13+25i / -3+2i / 58-i * 3+7i / 5-2i * 7+9i - 8+8i + -3+2i - 5+6i
```

Example of output
```
15-15i
-581+554i
1.5-6.5i
-14.85-10.16i
```

Explanation of the first output
We are dealing with additions and subtractions of complex numbers, so we separate the number into real and imaginary parts. In the real part, we have 5+2--8 = 15 and in the imaginary part, we have 7-3-19 = -15. So we get 15-15i.