A	T9	
	Code Name	`t9`
	Input	Standard Input
	Output	Standard Output
	Time Limit	2 second
	Memory Limit	256 megabytes

Problem Statement

Dr. Nazim is a nostalgic person. He enjoys using old technologies that he discovered during his adolescence in the 90's of the previous century. One of his favorite technologies is the old phone Nokia 3310. He used to send messages using the old keyboard where you have to click a number several times to see the letter you want to print on the screen. A long period using this kind of keyboard led him to memorize several numbers representing words. He can even convert any word to a number while typing text to his friend Dr. Yessine. This is Dr. Nazim's phone.

Can you write a program that can think as Dr. Nazim by converting each expression to a number?

Programming League National 2020

Input Format

The first line contains a single integer $N(1 \leq N \leq 100)$ which is the number of test cases. Each test case consists of one line containing an expression based on simple characters (letters) or spaces.

Output Format

For each test case print the corresponding number.

Please note that the output for upper and lower case are the same.

Sample Input	Sample Output
2 Hello good morning	4433555555666 466666630666677766444664

B IP Checking

Code Name	`ipchecking`
Input	Standard Input
Output	Standard Output
Time Limit	2 second
Memory Limit	512 megabytes

Problem Statement

An IP address is a 32 bit address formatted in the following way

$$a.b.c.d$$

where a, b, c, d are integers each ranging from 0 to 255. Now you are given two IP addresses, first one in decimal form and second one in binary form, your task is to find if they are the same.

Input Format

Input starts with an integer $T(1 \leq T \leq 100)$, denoting the number of test cases.

Each case starts with two lines. First line contains an IP address in decimal form, and second line contains an IP address in binary form. In binary form, each of the four parts contains 8 digits. Assume that the given addresses are valid.

Output Format

For each case, print the case number and *"Yes"* if they are the same, *"No"* if otherwise.

Sample Input	Sample Output
2 192.168.0.100 11000000.10101000.00000000.11001000 65.254.63.122 01000001.11111110.00111111.01111010	Case 1: No Case 2: Yes

Programming League National 2020

C Memory Usage

Code Name	`memoryusage`
Input	Standard Input
Output	Standard Output
Time Limit	3 second
Memory Limit	1024 megabytes

Problem Statement

Ahmad is a system administrator. He handles Linux server. Ahmad has a list of processes, their parent id, and their memory usage. Sometimes, he needs to know how much memory a process and all its descendant use. Given the list, and some process id, output the total amount of memory used by the process and its descendant combined all together.

Input Format

The input starts with two integers $n, q (1 \leq n, q \leq 10^5)$ where n is the number of process and q is the number of query. The next n lines each represents a process. Each line consists of three integers $a, b, c (0 \leq a, b, c \leq 10^5)$, where a is the *pid* or *process id*, b is the *ppid* or *parent process id*, and c is the *memory usage*. The next q lines each represents a query. Each line consist of a single integer $p (0 \leq p \leq 10^5)$ which is the *process id*.

Output Format

For each p output the total amount of memory it and all its descendants are using

Note

In the second test case, the first query is for process 1 which is the parent of all other process, so the output is 50, the second query is for process 3 which is the parent of the other 3 process, so the output is 30.

Sample Input	Sample Output
5 2 1 0 10 2 1 10 3 2 10 4 3 10 5 4 10 1 3	50 30
5 5 1 0 12 2 1 22 4 1 92 100 4 98 101 4 92 1 2 4 100 101	316 22 282 98 92

D

Eid

Code Name	`eid`
Input	Standard Input
Output	Standard Output
Time Limit	3 second
Memory Limit	512 megabytes

Problem Statement

In a strange planet there are n races. They are completely different as well as their food habits. Each race has a food-eating period. That means the i^{th} race eats after every x_i *de-sec* (*de-sec* is the unit they use for counting time and it is used for both singular and plural).

The planet declared the de-sec as in which all the races eat together as *Eid*.

Now given the eating period for every races you have to find the number of de-sec between two consecutive Eids.

Input Format

Input starts with an integer $T(1 \leq T \leq 225)$, denoting the number of test cases.

Each case of input will contain an integer $n(2 \leq n \leq 1000)$ in a single line. The next line will contain n integers separated by spaces. The i^{th} integer of this line will denote the eating period for the i^{th} race. These integers will be between 1 and 10000.

Output Format

For each case of input you should print a line containing the case number and the number of de-sec between two consecutive Eids. Check the sample input and output for more details.

Tips from our friendly contest department's head: "please take extra care for the output of case 3"

Sample Input

```
3
3
2 20 10
4
3 10 5 6
14
7803 5480 5148 5575 3710 1823 2633 5486 4824 569 4108 4250 8034 1365
```

Sample Output

```
Case 1: 20
Case 2: 30
Case 3: 397345548877597784165436433799100 0
```

E Coin Change

Code Name	`coinchange`
Input	Standard Input
Output	Standard Output
Time Limit	3 second
Memory Limit	512 megabytes

Problem Statement

Suppose there are *5* types of coins: *50*-cent, *25*-cent, *10*-cent, *5*-cent, and *1*-cent. We want to make changes with these coins for a given amount of money.

For example, if we have *11* cents, then we can make changes with one *10*-cent coin and one *1*-cent coin, two *5*-cent coins and one *1*-cent coin, one *5*-cent coin and six *1*-cent coins, or eleven *1*-cent coins. So there are four ways of making changes for *11* cents with the above coins. Note that we count that there is one way of making change for zero cent.

Write a program to find the total number of different ways of making changes for any amount of money in cents. Your program should be able to handle up to 7489 cents.

Input Format

Input starts with an integer $T(1 \leq T \leq 225)$, denoting the number of test cases.

Each case of input will contain an integer $c(0 \leq c \leq 7489)$ which is the number for the amount of money in cents.

Output Format

For each input line, output a line containing the number of different ways of making changes with the above *5* types of coins

Sample Input	Sample Output
2 11 26	4 13

Programming League National 2020

F Guessing

Code Name	`decrypt`
Input	Standard Input
Output	Standard Output
Time Limit	3 second
Memory Limit	512 megabytes

Problem Statement

Number guessing is a popular game between elementary school kids. Teachers encourage pupils to play the game as it enhances their arithmetic skills, logical thinking, and following up simple procedures. We think that most probably, you too will master in few minutes.

Here's one example of how you too can play this game: Ask a friend to think of a number, Let's call it n_0. Then:

1. Ask your friend to compute $n_1 = 3 \times n_0$ and to tell you if n_1 is even or odd.
2. If n_1 is even, ask your friend to compute $n_2 = \frac{n_1}{2}$. If, otherwise, n_1 was odd, then let your friend compute $n_2 = \frac{(n_1+1)}{2}$.
3. Now ask your friend to calculate $n_3 = 3 \times n_2$
4. Ask your friend to tell you the result of $n_4 = n_3/9$. (n_4 is the quotient of the division operation. In computer lingo, '/' is the integer-division operator.)
5. Now you can simply reveal the original number by calculating $n_0 = 2 \times n_4$ if n_1 was even, or $n_0 = 2 \times n_4 + 1$ otherwise.

Here's an example that you can follow: If $n_0 = 37$, then $n_1 = 111$ which is odd. Now we can calculate $n_2 = 56$, $n_3 = 168$, and $n_4 = 18$, which is what your friend will tell you. Doing the calculation $2 \times n_4 + 1 = 37$ reveals n_0.

Input Format

Your program will be tested on one or more test cases. The first line contains the number $x(0 < x < 1000)$ of tests and each test case is made of a single positive number $n_0(0 < n_0 < 1,000,000)$

Output Format

For each test case, print the following line: `B<space>Q`, where B is either **'even'** or **'odd'** (without the quotes) depending on your friend's answer in step 1. Q is your friend's answer to step 4.

Sample Input	Sample Output
4 18 98 33 45993	even 9 even 49 odd 16 odd 22996

Programming League National 2020

G Decrypt

Code Name	`decrypt`
Input	Standard Input
Output	Standard Output
Time Limit	3 second
Memory Limit	256 megabytes

Problem Statement

DRM Encryption is a new kind of encryption. Given an encrypted string (which we'll call a *DRM message*), the decryption process involves three steps: **Divide**, **Rotate** and **Merge**. This process is Described in the following example with the DRM message "EWPGAJRB":

1. **Divide** - First, divide the message in *half* to "EWPG" and "AJRB", **Rotate** - For each half, calculate its rotation value by summing up the values of each character (A = 0, B = 1 ... Z = 25). The rotation value of "EWPG" is $4+22+15+6 = 47$.
2. **Rotate** each character in "EWPG" 47 position forward (wrapping from Z to A when necessary) to obtain the new string "ZRKB". Following the same process on "AJRB" results in "BKSC".
3. **Merge** - the last step is to combine these new stings ("ZRKB" and "BKSC") by rotating each character in the first string by the value of the corresponding character int the second. For the first position, rotating 'Z' by 'B' means moving it forward 1 character, which wraps it around to 'A'. Continuing this process for every character results in the final decrypted message, "ABCD".

Input Format

The first Line of the input file contains the number of test cases $T(1 \leq T \leq 500)$ followed by T lines containing string M.

All characters in the *DRM message* (M) to be decrypted are uppercase letters. length of M is ≤ 15000 and is *even*.

Output Format

For each test case, display the decrypted *DRM message*.

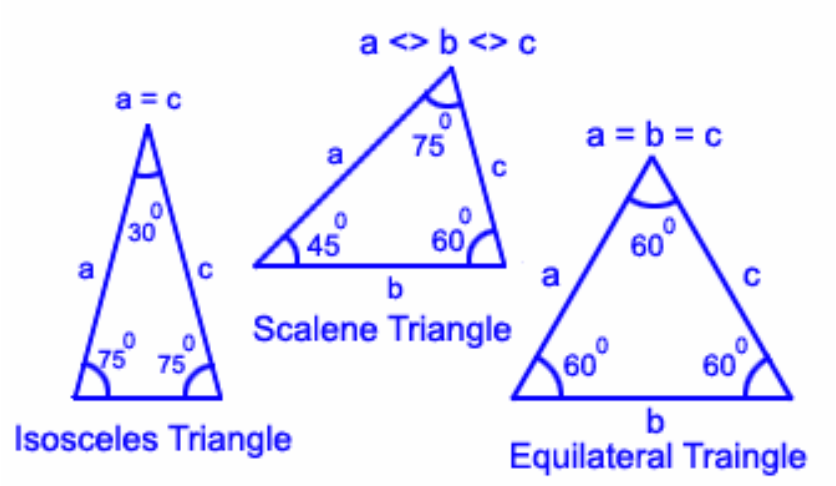
Sample Input	Sample Output
1 EWPGAJRB	ABCD

H Triangle

Code Name	`triangle`
Input	Standard Input
Output	Standard Output
Time Limit	2 second
Memory Limit	256 megabytes

Problem Statement

A triangle is a polygon with three edges and three vertices. The three edges must form a closed region. Thus, not any given three edges will form a triangle. Triangles can be categorized depending on the values of the edges of a valid triangle. Now you are required to determine the type of a triangle.



Input Format

The first line of input will contain a positive integer $T(1 \leq T \leq 200)$, where T denotes the number of test cases. Each of the next T lines will contain three 32-bit signed integer.

Output Format

For each case of input, there will be one line of output. It will be formatted as: `<triangle type>` in every line

Where `triangle type` is the type of the triangle. Triangle type will be one of the following, depending on the values of the three edges:

- **"Invalid"** - The three edges can not form a triangle
- **"Equilateral"** - All three edges of valid triangle are equal
- **"Isosceles"** - Exactly two of the edges of a valid triangle are equal.
- **"Scalene"** - No pair of edges are equal in a valid triangle.

Sample Input	Sample Output
4 1 2 5 1 1 1 4 4 2 3 4 5	Invalid Equilateral Isosceles Scalene

Rectangle

Code Name	`rectangle`
Input	Standard Input
Output	Standard Output
Time Limit	5 second
Memory Limit	1024 megabytes

Problem Statement

Given n points on the XY plane, count how many regular rectangles are formed. A rectangle is regular if and only if its sides are all parallel to the axis.

Input Format

The first line contains the number of tests $t(1 \leq t \leq 10)$.

Each case contains a single line with a positive integer $n(1 \leq n \leq 5000)$, the number of points.

There are n lines follow, each line contains 2 integers $x, y(0 \leq x, y \leq 10^9)$ indicating the coordinates of a point.

Output Format

For each test case, print the number of regular rectangles found.

Sample Input	Sample Output
2 5 0 0 2 0 0 2 2 2 1 1 3 0 0 0 30 0 900	1 0

J

Decoder

Code Name	`decoder`
Input	Standard Input
Output	Standard Output
Time Limit	5 second
Memory Limit	512 megabytes

Problem Statement

Ting and Jing are relatives that live in different towns. As they inhabit a rural area, it is very difficult for them to keep in touch. One way they found to overcome their communication problem was to send a line through their parents that used to visit each other.

The point is that Ting and Jing did not want that their parents read their messages, and they decided to create a secret code for the messages. The code is not very sophisticated, but you should keep in mind Ting and Jing are just children.

In general, the meaning of a message is based on a letter of each word, in a way that they will form a message with the first letter of the first word, the second letter of the second word and so on. If a word does not have enough letters, the following word should be used. For example, if you are analyzing the third word, you should consider its third letter, but if it just has two letters, then you should try to form a decoded word with the third letter of the fourth word.

When the end of a line is reached, you should finish the current decoded word and should start to form another one from the first letter of the first word in the next line.

Your task is to translate a message according to Ting and Jing's secret code.

Input Format

The first line of input gives the number of cases, $T(1 \leq T \leq 30)$, then there is a blank line before the first test case.

Each test case represents a message, which is composed by $1 \leq N \leq 100$ lines and each line is composed by $1 \leq M \leq 30$ words. Two words in the same line are separated by one or more white spaces.

A word is formed by the letters A-Z and a-z and has at most 30 letters. The only symbols that appear in the input are the alphabetic letters and white spaces. There will be a blank line between two test cases as well.

Output Format

For each test case print each word of the decoded message, one per line (look the sample output for the exact format). You must print a blank line between each test case.

Sample Input	Sample Output
2 Hey good lawyer as I previously previewed yam does a soup First I give money to Teresa after I inform dad of your horrible soup	How are you Fine and you