Problem A - Let’s start the contest

Passwords provide the first line of defence against unauthorized access to your computer. The stronger your password, the more protected your computer will be from hackers and malicious software. You should make sure you have strong passwords for all accounts on your computer. If you're using a corporate network, your network administrator might require you to use a strong password.

What makes a password strong (or weak)?

A strong password:

1. Is at least eight characters long.
2. Does not contain your user name, real name, or company name.
3. Does not contain a complete word.
4. Is significantly different from previous passwords.
5. Contains characters from each of the following four categories:

Character category	**Examples**
Uppercase letters	A, B, C
Lowercase letters	a, b, c
Numbers	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Symbols found on the keyboard (all keyboard characters not defined as letters or numerals) and spaces	` ~ ! @ # $ % ^ & * ( ) _ - + = { } [ ] \ | : ; " ' < > , . ? /

The above sentences were gathered from Microsoft website for creating a strong password.

But honestly, now, we aren’t concerned about your strategies for choosing passwords in your life! We just want to start the contest as simple as possible to prevent some bad memories occur again.

So for simplicity we just want you to check if some passwords has the condition 5 or not. For more simplicity you can restrict the symbols to these set:

`SYM = {!, @, #, $, %, ^, &, *, (, )}`

Note that ‘,’ and ‘{‘ and ‘}’ are not considered as SYM members.

Given a string S, you should print the security state of the password.

The security state of a password defined in this way, If given password has exactly D type of characters category its security state for:

- D = 1 is ‘`very bad`’
- D = 2 is ‘`bad`’
- D = 3 is ‘`good`’
- D = 4 is ‘`overkill`’

Input

The first line of the input contains an integer t ($1 \le t \le 1000$), the number of test cases.

Each test case has one string S, which consists of lowercase and uppercase English alphabets ('a'...'z' 'A'...'Z'), digits('0'...'9') and members of SYM.

Length of S is less than 50.

Output

For each test case print the security state of password.

Sample Input	Sample Output
4 SaateConteste8Aban BarcaPerspolisRealEsteghlalNavadVoting numberofqualifiedteamstowfinpreviousyear @SBUContest2014	good bad very bad overkill

Problem B - List Analysis

Speed violation detection is not a simple task. After the speed of a vehicle is determined, there are several rules that should be considered before ticketing the car.

1- The speed limit in different routs may be different. For example the speed limit in Modarres highway is 80km/h but in Hemmat highway is 100km/h

2- The speed limit in different hours may differ. For example in Hemmat highway it is 70km/h after 11:00 pm till 5:00 am

3- Also the speed limit for different types of vehicles is different. For example in highways the speed limit for heavy vehicles is 30km/h lower than the other vehicles.

Having a list of passing vehicles in different routs of the city and a set of rules and information about the speed limits and vehicle types, you are asked to write a program that determines the vehicles that should be given a ticket for speeding.

You can assume that there will be enough information to determine correctly.

Input

The input consists of 3 parts.

First part consists of several lines, each line describing a speed limit rule for a route in a specific time (times are inclusive). Each line is in the following format:

`<highway name> : <start time>-<end time> : <vehicle type> : <speed limit>`

Second part is a list of vehicles' types as follows

`<license plate> : <vehicle type>`

And the third part is the list of passing vehicles from different routs in different times. Each line is as follows

`<license plate> : <speed> : <passing time> : <highway name>`

Remember, if two rules overlap, in the overlapped section <u>only</u> newer rule will apply.

Also consider:

```
<highway name> : (a~z|A~Z|_)*
<time> : DD:DD:DD (00:00:00 ~ 23:59:59)
<vehicle type> : heavy | light
<speed limit> & <speed> : unsigned integer less than 999
<license type> : DD-<letter>-DDD-DD (12-alef-245-22 | 22-dal-143-19)
<letter> : alef | beh | peh | teh | jim | dal | sin | sad | ta | ein | ghaf |
kaf | lam | mim | noon | vav | heh | yeh
```

There might be more or none spaces before and after "**:**" in order to align the input.

There will be less than 51 highways, 2001 highway rules, 10001 licenses and 100001 queries.

Output

Output the list of the passing vehicles which have violated the speed limit. (their speed is bigger than speed limit)

The output list should be sorted alphabetically increasing, according to their license plate strings. If there are more than one entries of a vehicle, they should be sorted according to their passing time. If same passing times exist, they should be sorted according to their highway names. If same highway names exist, they should be sorted decreasingly according to their speed.

The list should be aligned using extra spaces using the least possible spaces. Licenses have to be left-aligned and speeds have to be right-aligned. There must be at least one space before and an at least one space after "`:`" (not the ones in time)

Sample Input

```
modares     : 06:00:00-22:00:00 : light : 80
modares     : 22:00:00-06:00:00 : light : 70
modares     : 00:00:00-23:59:59 : heavy : 60
shahid_sadr : 00:00:00-23:59:59 : light : 100
shahid_sadr : 00:00:00-23:59:59 : light : 70
22-alef-234-11 : light
11-dal-239-22  : heavy
12-vav-223-33  : light
99-noon-453-11 : light
11-dal-239-22  : 160 : 09:23:09 : modares
99-noon-453-11 : 75 : 04:23:34 : shahid_sadr
11-dal-239-22   :  75   : 23:27:29 : modares
11-dal-239-22  :  60 : 07:23:09 : modares
```

Sample Output

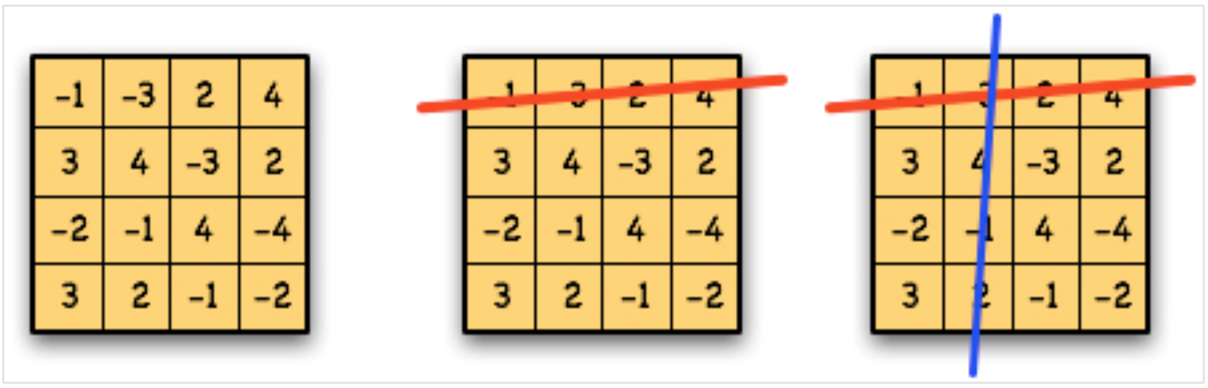
```
11-dal-239-22  : 160 : 09:23:09 : modares
11-dal-239-22  :  75 : 23:27:29 : modares
99-noon-453-11 :  75 : 04:23:34 : shahid_sadr
```

Problem C - Grid Game II

Atefeh and Behrad both have lots of saffron but want more. They decide to play the following turn-based game.

They fill an n x n grid M with random integers. Atefeh begins the game by crossing off an uncrossed row i of the grid. Now it's Behrad's turn and he crosses off an uncrossed column j of the grid. At the end of Behrad's turn, Atefeh takes the number saffrons in the ith row and jth column of M, call this value M(i, j), from Behrad. (If M(i,j) is negative, then Atefeh gives |M(i, j)| saffron to Behrad.) The game continues alternating turns from Atefeh to Behrad until the entire board is crossed off.

What is the largest amount of saffrons that Atefeh can win from Behrad (or least amount to lose if she cannot win) if both Atefeh and Behrad play optimally?



The beginning of a game between Atefeh (red) and Behrad (blue).

Input

The first line of the input contains an integer t (1 ≤ t ≤ 20), the number of test cases. Each test case starts with n (1 ≤ n ≤ 15), the size of the grid. Then follow n lines containing n numbers separated by spaces describing M. We call the j-th number on i-th line M(i, j) (-1000 ≤ M(i, j)≤ 1000).

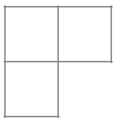
Output

For each test case, print the largest amount of saffron that Atefeh can win from Behrad. If she cannot win, print the negative number indicating the minimum number of saffron she loses.

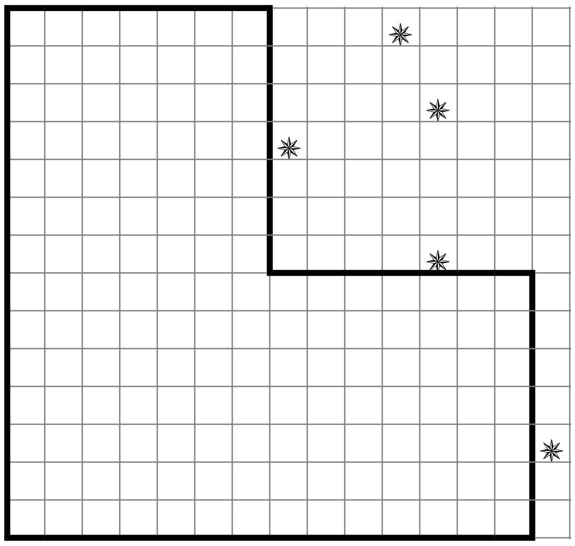
Sample Input	Sample Output

3
2
10 10
-5 -5
2
10 -5
10 -5
2
10 -5
-5 10
	5
	5
	-10

Problem D - Largest Villa



Having a grid based map, we would like to build a villa that its plan is in the shape of below or any rotation of it (a three quarter of a square)
On the other hand, there are some trees on the map which we certainly don't want to cut down. Your task is to find the biggest area (which is like the shape above) that has no trees in it.



Input

The input consists of several test cases (at most 30 test cases). Each test case starts with 3 integers, M, N and T. The first two are the dimensions of the field, and the third integer is the number of the trees.
This line is followed by T lines, each having 2 integers, the coordinates of a tree (indexed from zero).

$2 \le M, N \le 4000$

$T \le \frac{M.N}{2} - 1$

Output

For each test case, out put the maximum possible area for the desired villa.

Caution: Huge input!

Sample Input	Sample Output

15 14 5
10 0
11 2
7 3
11 6
14 11
	147

Problem E - GCD

Shamir has recently learnt about Greatest Common Divisor between numbers. He counted there are 7 pairs of numbers between 1 and 6 (inclusive) that their GCD(Greatest Common Divisor) equals 2: {(2,2),(2,4),(2,6),(4,2),(4,6),(6,2),(6,4)}. He wants to count all pairs of numbers (X,Y) such that $x_1 \le X \le x_2, y_1 \le Y \le y_2$ and $\gcd(X,Y) = d$.

- The greatest common divisor (gcd) of two or more integers (when at least one of them is not zero), is the largest positive integer that divides the numbers without a remainder. For example, the GCD of 8 and 12 is 4.

Input

The input contains several test cases. First line of input contains an integer T indicating number of test cases and is followed by T lines each containing 5 integers x_1, x_2, y_1, y_2 and d describing the test-case.

$1 \le x_1, x_2, y_1, y_2, d \le 1000000, x_1 \le x_2, y_1 \le y_2$

Output

For each test case print total number of pairs explained above on a single line.

Sample Input	Sample Output

3
1 6 1 6 2
1 9 3 11 3
1 10 1 10 1
	7
	7
	63

Problem F - Remote Bike

Shamir's dad bought a Remote Bike for him. Shamir can make this Bike do 4 tasks: accelerate, decelerate, jump, and idle. Shamir was playing with his bike the other day and all of a sudden, he saw bunch of holes. He was wondering how hard can it be to make the bike go over the holes without falling into any of them?
Shamir is a great programmer but he's busy playing with his bike, can you write a program to help him?

The Program:

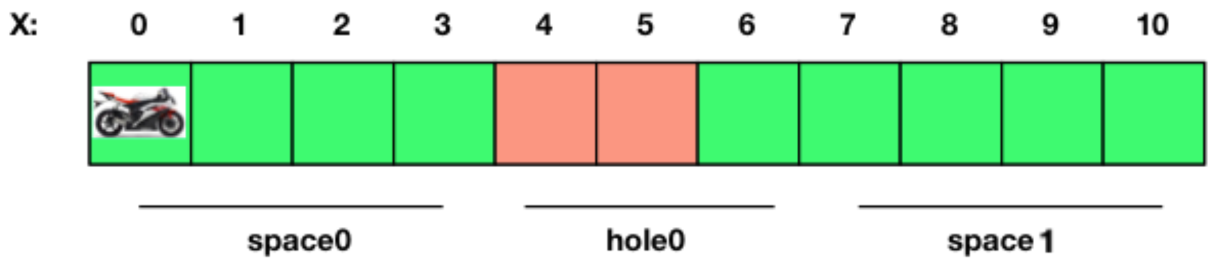
Write a program which tells bike what to do!

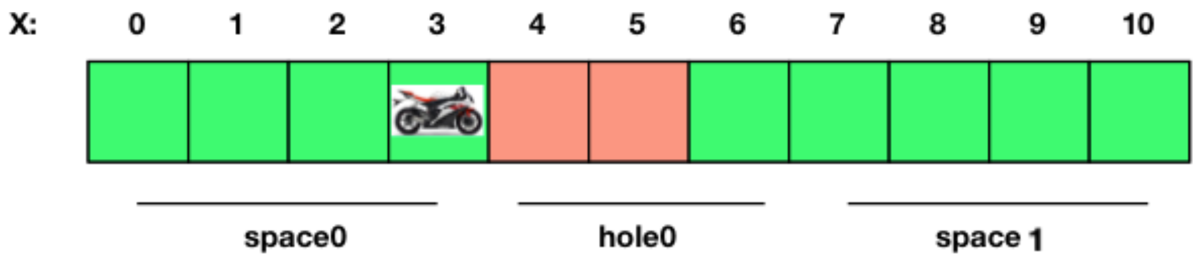
- Bike is moving on *X* axis and starts it's movements from $x=0$.
- Bike's remote accepts 4 instructions: "`Acc`", "`Dec`", "`Idl`" and "`Jmp`".
  - "`Acc`": increases speed of bike $1m/s$ and bike moves s meters. (s = bike's speed)
  - "`Dec`": decreases speed of bike $1m/s$ and bike moves s meters.
  - "`Idl`": does nothing; bike keeps moving s meters.
  - "`Jmp`": makes bike jump; and bike moves s meters.
- Bike already has a speed; we call it s_0 (initial Speed).
- Bike can't go backwards; speed can't be negative. ($s \geq 0$)
- You have to set all instructions, before bike starts moving; you tell bike what to do before it starts moving, once it starts moving, it will run your instructions one by one.
- If bike reads an instruction, it will run it immediately. What does it mean? It means for example if bike is at position x=5 and s=4 and instruction is "Acc", speed will increase to 5 then bike will move, it will result in: x=10 and s=5.
- Shamir's bike is brand new, he doesn't like to Jump a lot. He wants to go over all holes with minimum number of jumps.
- Do not fall into a hole.
- Each hole has different length.
- There are some spaces before, between and after holes.
- Shamir wants to stop his bike($s=0$) after going over the last hole in the field. (bike has to stop in space after the last hole).
- Bike's speed can't exceed 100.

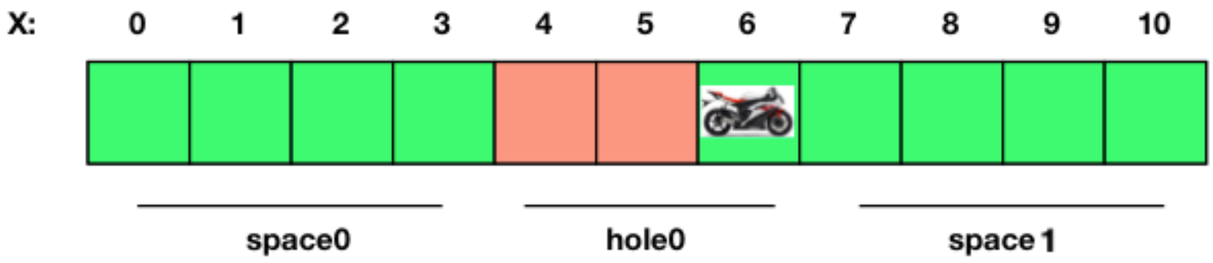
Sample:

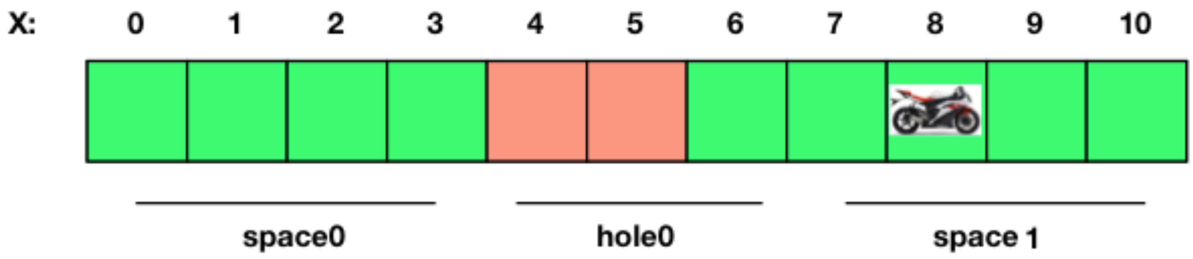
Bike has $s_0 = 3$:

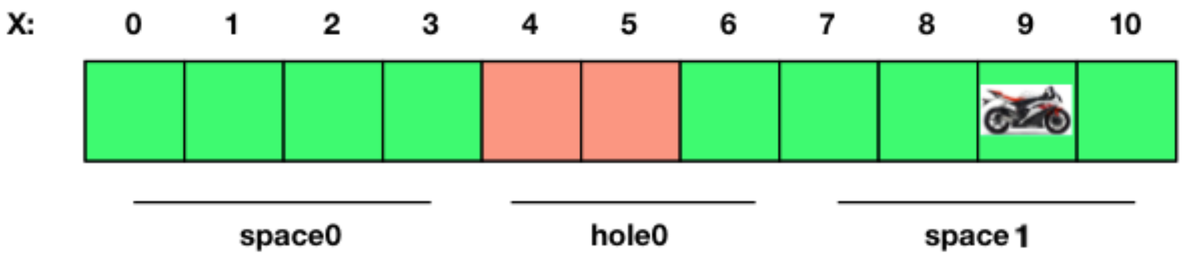
Instructions are: `"Idl", "Jmp", "Dec", "Dec", "Dec"`

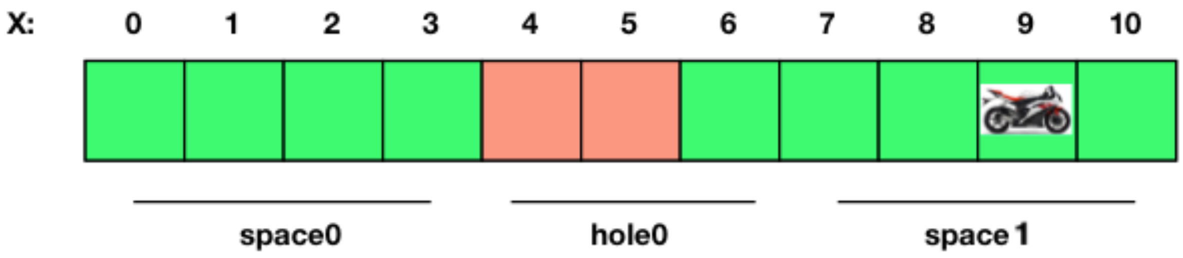

X: 0 1 2 3 4 5 6 7 8 9 10
space0
hole0
space1


X: 0 1 2 3 4 5 6 7 8 9 10
space0
hole0
space1


X: 0 1 2 3 4 5 6 7 8 9 10
space0
hole0
space1


X: 0 1 2 3 4 5 6 7 8 9 10
space0
hole0
space1


X: 0 1 2 3 4 5 6 7 8 9 10
space0
hole0
space1


X: 0 1 2 3 4 5 6 7 8 9 10
space0
hole0
space1

Input

Input starts with an integer T on a single line indicating number of test cases. Each test-case is described as follows:

Test case contains two lines. First line contains two integers N, S indicating number of spaces + holes and initial speed. Other line contains N integers $space_0, hole_0, space_1, hole_1, ..., space_{\frac{n-1}{2}}$ length of spaces and holes separated by white spaces.

N is odd, $0 < N < 12$, $0 \le S \le 100$, $space_i, hole_i > 0$, $\sum space_i + \sum hole_j \le 10000$

Output

For each test-case:

Step one:

Print in a line: "START"

Step two:

Print instructions to lead the bike.

If there's more than one solution, each will do.

Step three:

Print in a line: "END"

If there are no solutions, print in a single line "Sorry Shamir!" instead of three steps.

Sample Input	Sample Output

3
3 3
4 2 5
3 0
7 2 4
3 0
7 2 3
	START
	Idl
	Jmp
	Dec
	Dec
	Dec
	END
	START
	Acc
	Acc
	Acc
	Jmp
	Dec
	Dec
	Dec
	END
	Sorry Shamir!

Problem G - Not So Rapidshare

Nonlogius and Linearius are two brothers developing the next big file hosting service. They are currently implementing the upload request handling module. They want the module to be able to do the following tasks:

1. Receive new files to be processed and add them to the line of waiting files.
2. Pass the first file in the waiting line to the file saving module and report ID of that file.
3. Report size of the biggest file waiting to be processed.
4. Report sum of the sizes of all files waiting to be processed.

Unfortunately, the brothers could not implement this module efficiently; therefore, they have asked you to help them in doing this hard job.

Input

The input will have an integer *TC* ($TC \le 10$), the number of test cases, in the first line. TC test cases will follow. First line of each test case will contain a single number *Q* ($1 \le Q \le 2*10^6$), the number of queries to be processed by the module. Each of the following *Q* lines will contain description of a query. Each query will start with a single character that is equal to “R”, “P”, “M”, or “S”. If the character is equal to “R” it will be followed by two integers *ID* and *Size* ($0 \le ID < 10^9$, $1 \le Size \le 10^9$). Description of each query is as follows:

“R”: Receive a new file and its’ ID and size.

“P”: Pass the first file in the waiting line to the other module, and report its’ ID.

“M”: Report the maximum file size in the waiting line.

“S”: Report sum of the sizes of all files in the line.

File IDs are guaranteed to be distinct in each test case.

Output

For each query of type “P”, “M”, or “S”, print the requested value in a single line. It is guaranteed that for all queries of type “P” or “M”, there will be at least one file in the waiting line. Note that if all queries are of type “R” in a test case, you don’t need to print anything for that test case.

Caution: Huge input/output!
Consider: using faster methods (e.g. scanf/printf).

Sample Input	Sample Output
3 1 R 131452 99999 7 R 7000 1100 S R 3200 900 M P M P 12 R 123 250 R 1543 1 M S P P S R 131452 1500 R 11111 12000 M P S	1100 1100 7000 900 3200 250 251 123 1543 0 12000 131452 12000

Problem H - Minimum Cost

There are three types of transportations in the city;
First is the taxi, which its cost, somehow depends on the distance you've got a cab for.
Second type of transportation is the bus, which costs 1000 Rials to use for each stop.
And the third is the metro which cost 5000 Rials for entering the subway and using as many lines as you want, as long as you have not gotten out of the subway. The metro lines are always bidirectional.
Now, knowing the availability and costs of taxi routes, we want you to determine the cost of travelling for a series of source-destination pairs.

Input

The input contains two parts. In the first part, city's transportation routs are suggested. Each connection is shown in a line using three values. The first two values are the start and end point's names. The third value can be either `'M'`, `'B'` or an integer. Which `'M'` represents a metro connection, `'B'` a bus connection and in case of integer, it shows the taxi cost for that route. in the second part, there are several paired values, the names of source and the destination points. There will be at most 1000 nodes in the city.

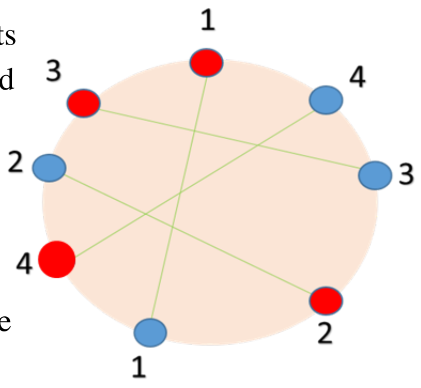
Output

For each pair, print source, destination and minimum cost of the travel, separated by a space. If there is no way to get to the specified destination from the specified source, using these three kind of transportations, output "`should walk!`" instead of the minimum cost.

Sample Input	Sample Output
tajrish gheitarie M gheitarie sadr M sadr gholhak M gholhak shariati M shariati mirdamad M mirdamad haghani M tajrish haghani B haghani tajrish B tajrish enghelab B enghelab tajrish B tajrish azadi B azadi tajrish B azadi enghelab B enghelab azadi B enghelab tajrish 20000 tajrish azadi 20000 enghelab ghazvin 15000 tajrish ghazvin ghazvin tajrish	tajrish ghazvin 16000 ghazvin tajrish should walk!

Problem I - Intersections

There are N red points numbered from 1 to N, and also N blue points numbered from 1 to N around a circle. We want to connect each red points to its corresponding number in blue points and then counting number of all intersections which occur. Given the order of blue and red points around the circle, your task is to count number of all intersections after connecting points in a mentioned way.



You can assume that no 3-pair of points intersect at the same position.(i.e you should count all pairs of intersecting pairs separately)

Input

input contains several test cases. Each test case is composed by three lines. The first line contains an integer *N*, indicating number of red and blue points $1<N<100000$. The second line contains a string S, consisting of 'R' and 'B' characters that are clockwise order of blue and red points. The third line contains $2*N$ integers $p_0, p_1, ..., p_{2*n}$ separated by spaces. p_i is the number of each corresponding character in second line. so the i-th point around the circle (from wherever you start, in clockwise order) has the number p_i and color s_i .

You can assume that input is valid, i.e. there are exactly N of 'R' and N of 'B' characters in S. Also each number p_i comes exactly twice and color of same numbers are different.

The input ends with a single zero.

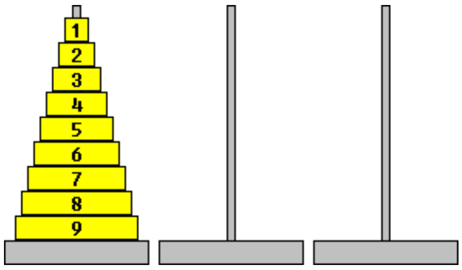
Output

For each test case in the input, print the number of intersections as mentioned above.

Sample Input	Sample Output
4 RBBRBRBR 1 4 3 2 1 4 2 3 3 RRBBRB 1 2 2 1 3 3 0	5 0

Problem J - Hanoi Towers in Detail

Consider the famous Hanoi Towers puzzle. There are pegs 1, 2, 3 and a pyramid of N discs of different sizes, initially collected at some peg - for example 1 - with the smallest disc at the top and the largest one at the bottom. The discs should be transferred to another peg - for example 3 - using the intermediate peg - for example 2 - by moving one disc at a time and placing a smaller disc either on top of a larger one or on an empty peg.



Given the number of discs *N*, and the indexes of the initially full *S*, and the destination pegs *D* and a disc index *i(1<=i<=N)*, you should print three numbers indicating that how many times the specified disc *i* was moved to pegs 1, 2 and 3 respectively, if you move all discs from peg S to peg D optimally (i.e. using minimum number of moves). As these three numbers can be very large, you should print them modulo 1000000007.

Input

The first line of input contains a single integer *T*, the number of test cases to follow. The only line of each test case contains four integers, *N, S, D* and *i* as mentioned above.

$T \leq 100000$, $1<=N<=1000000000$ *and* $S \neq D$.

Output

for each test case print three space separated numbers, as explained above.

Sample Input	Sample Output

5
3 1 3 1
3 1 3 2
6 3 2 6
5 2 1 2
2 2 3 1
	1 1 2
	0 1 1
	0 1 0
	3 2 3
	1 0 1

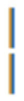
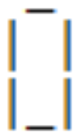
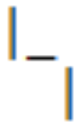
Problem K - Milliseconds Clock

When you are working with synced cameras, like any other field, you need some extra utilities for testing purposes. One of these utilities, is a simple milliseconds clock. Why? Because sometimes you need to test the sync of the two cameras using a unique reference. Both cameras are put in front of the clock and start getting single shot frames.

Assuming the exposure time of the camera is 1 millisecond, the shutter may be open during two different milliseconds of the clock, which makes the taken frame's captured clock state, undecidable.

Your task is to determine what values where possible in a shown state of a three digit seven segment.

For example if a one digit seven segment was captured showing below states:

Shown	Possibilities
	1
	0,1,7
	impossible

Input

The input consists of several test cases and ends with end-of-file

Each test case is represented in three lines and there is a blank line after each test case.

Each seven segment is written in a 3*3 character box, which makes each test case a 3*9 character box. Vertical segments will be given with "|" and horizontal segments will be given with "_".

Output

For each test case, output all the possibilities one after another, in increasing order, separated by commas.

If there is no possible value, output "`impossible`"

Sample Input

```
 _     _
| |  | _|
|_|  ||_

 _     _
 _||_||_
 _|  | _|

 _
|_   |  |
|_|  |  |

 _
  |  |  |
  |  |  |

 _
|_|  |  |
|_|  |  |

 _
|_|  |  |
 _|  |  |

 _     _
 _||_| _
 _|  | _|
```

Sample Output

```
012, 112, 712
115, 145, 315, 345, 715, 745
511, 611
111, 711
011, 111, 211, 311, 411, 511, 611, 711, 811, 911
111, 311, 411, 511, 711, 911
impossible
```