


ACM ICPC - Sharif Internet Contest & Internal Selection Contest of Sharif

7 Azar 1387 - 27 Nov. 2008

Table of Contents

Note:

Your codes must read from standard input (`stdin`) and write to standard output (`stdout`).

Problem A: A Decimal Number that Fits in a 32-bit Integer Type is Given

[This part is intentionally left blank]

Input

First line of Input contains $\mathbf{1} \leq \boldsymbol{T} \leq \mathbf{100}$ that is number of the tests. Then, in each of the following T lines, comes an integer number that fits in an `int` type variable of C/C++ and Java.

Output

See the sample below.

Sample Input	Sample Output
5 1 29 100 1534 9999905	1 11 1 13 50

Hint

Since $1 + 5 + 3 + 4 = 13$ and $9 + 9 + 9 + 9 + 9 + 0 + 5 = 50$, the problem asks you to write the summation of all decimal digits for each given number.

Problem B: **Bold**, *Italic*, Underline

, and <u> are 3 famous HTML tags that make the enclosed text (what appears between <x> and </x>) respectively bold, italic, underline. For example Hello creates **Hello** and <u>help me</u> crates Help me.

Parisa is addicted to XHTML in which the tag is commonly used (instead of the above tags) and the above behaviors (like making bold or italic or underlined) are described in the tag's `style` attribute.

Parisa replaces the old tags with the new alternative ones, when she sees an old tag. The old tags and their new alternatives are shown in the table below.

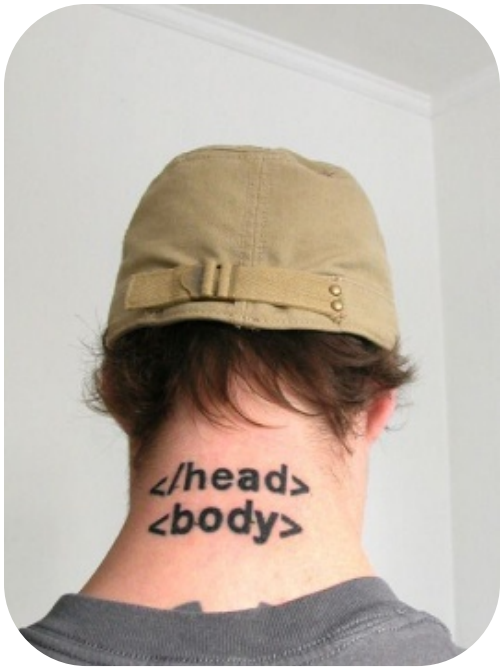
Before: Tag (and example)	After: Tag (and example)
text	text
text	text
<u>text</u>	text

Notice that there are only two blank spaces in the new format: first after `span` and before `style`, second after the colon inside style attribute.

After these modifications, Parisa likes to merge adjacent span tags with their attributes. See table below:

Before	text
After	text

Notice that there is a blank space after `X;` and before `Y;` in the new (After) format, and X is supposed to be lexicographically less than Y (otherwise it would be `"Y;  X;"`). Parisa applies this procedure as many times as there are two adjacent tags remained. If X (that might be a set) and Y (that might be a set as well) are equal or have any attribute in common, that common attribute would be written only once.



Given an old style (using , , <u>) HTML text, you are asked to apply what Parisa expects and the print out the result!

Input

First line of input has number of test cases $\mathbf{1} \leq \boldsymbol{T} \leq \mathbf{100}$. Then in any of next $\boldsymbol{T}$ lines one string comes per line. It's guaranteed that this string is made of at most 80 lowercase letters enclosed by some of the mentioned 3 tags (maybe recursive). There would be no other tags and the input string is valid i.e. text will not appear and all opening tags are closed and all closing tags have been opened before. No tag will be empty (i.e. is not valid).

Output

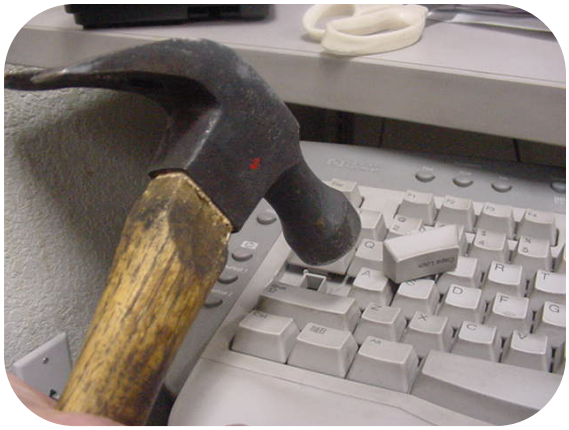
Write the expectation of Parisa for each input strings one per line!

Sample Input	Sample Output
6 hi hi aa <u><u>me</u></u> xy <u>x</u>	hi hi <span style="font-we↵ ight: bold;">a<span style="font-weight: bol↵ d;">a me x<span style="fon↵ t-style: italic;">y <span style=" font-style: italic; font-weight: bol↵ d; text-decoration: underline;">x

Note that sample output should have 6 lines in fact, but running out of space here, some lines are wrapped (identifiable by ↵ sign). For example, the last line of your answer (for 6th input string) must contain something like … `font-weight: bold; text-decoration` …

Problem C: !cAP!sLo!!!CK

"*Children are cute! We like to play with them most of the times but not when typing an important email!*" Said Nima's grandpa with smile and Nima wondered what had been that email!



Last week, when Nima was typing a code, his curious cousin Saeed was disturbing him by hitting random keys of keyboard in the mean time! Pulling the kid as far as possible, Nima restricted his access range to CapsLock key only; so when Nima typed **hello** (**h**, then **e**, …, then **o**) and Saeed hit the CapsLock once after **h** and later after the second **l**, **hELLo** appeared on the screen. In order to make the things funnier, Saeed decided to push a **!** key too, when he hit the CapsLock! Thus if Nima willed to write **hello** in the above example, what would appear at the end was **h!ELL!o**.

Given the appeared text, you are asked to write what Nima wanted to write!

Input

First line of input has number of test cases $\mathbf{1} \leq \boldsymbol{T} \leq \mathbf{1000}$. Then in any of next $\boldsymbol{T}$ lines one string comes per line that is what showed on the screen at the end. Each string has at least one and at most 80 characters of lowercase, UPPERCASE or ! sign only.

You should assume that in the beginning of each test, the CapsLock key is initially off.

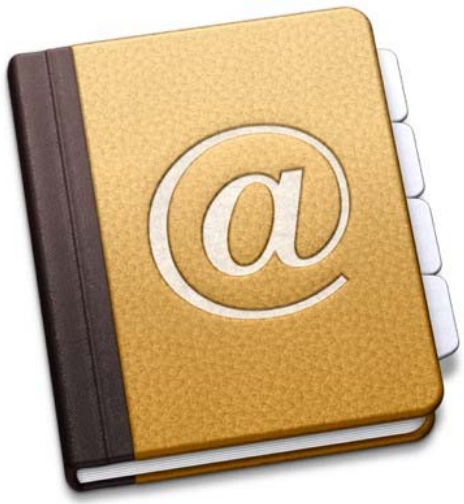
Output

Write the desired string (that is made of letters only) for each input case, one per line in the order they are given.

Sample Input	Sample Output
3 h!ELL!o !cAP!sLo!!!CK! !!!s!!!a!!!E!!E!d!b!as!k!on!dI!g!E!	Hello CapsLock SaeedBasKonDige

Problem D: DMail and its Address Book

"*Oops! This private email is sent to ma**safar**i@DMail.com (Dr. Safari, my supervisor in the department), instead of mahdi**safar**nejad@DMail.com (my friend), after typing **safar** in DMail's **TO:** field and hitting **Send** button immediately!*" said Aideen with grieve. :(



DMail (like the famous GMail!) is not idiot itself. The policy, by which it replaces email address, when typing a part of it and hitting enter, is based on recent sent mails. For instance, in the above case, the last email of Aideen to *Dr. Safari* had been sent later (more recent) than his last email to *Mahdi Safarnjead*.

For the sake of simplicity, here we just talk about email addresses (and not display names or anything else). Also we assume that all emails addresses end with ***@DMail.com*** and thus we discard the domain name -- just *masafri* and *mahdisafarnejad* in the above case is considered.

Speaking more clearly, we can assume DMail stores a precedence table for each user (like Aideen). Initially, when someone registers an account, DMail allocates an empty table for him. After that, at each step user types a string S (like ***safar*** in above example) and hits enter. Then, DMail seeks for the email addresses which contain the given string as a substring or whole. One of the following cases takes place:

1. If only one address had the given string, that address will be returned.
2. If more than one email address matched (like the example above), DMail returns the one with higher precedence and in the case of a tie, the one that comes lexicographically sooner.
3. Otherwise, if S is not a substring of any of the records of table, then S will be added to the precedence table with precedence value equal to 1.

In the first two cases in which an email address is reused, the precedence value of that email address in the table will be multiplied by two.

Now, given the order of sent emails, you are asked to act as DMail.

Input

$\mathbf{1} \leq \boldsymbol{T} \leq \mathbf{100}$, the number of test cases comes. After that, comes $\boldsymbol{T}$ block, describing one test case each. Each block begins with a number $\mathbf{1} \leq \boldsymbol{N} \leq \mathbf{1000}$ that is the number of emails user sends. Then in the following $\boldsymbol{N}$ lines, comes one string (entered by user) per line.

It is guaranteed that all given strings are between $\mathbf{1}$ and $\mathbf{50}$ lowercase letters of English. No more than $\mathbf{126}$ emails will be sent to a particular recipient. All email

Output

For each string, write the suggestion of DMail. That is, if no email address in the table contained the given string, return that and add it to the table (with precedence value equal to 1). Otherwise, pick one (of the already added into table) email address that has the given string and among all, has highest precedence and among all, comes sooner in dictionary.

Leave a blank line after each block.

Sample Input	Sample Output
3 1 hamid 6 aideen nimaee i aee ma i 4 ali alireza reza ali	hamid aideen nimaee aideen nimaee nimaee nimaee ali alireza alireza

Problem E: Empire of Extant AnacondaTigers

Long long time ago, there was a hybrid animal called AnacondaTiger! It had the head of a Tiger and the body of an Anaconda (see the photo in right).

AnacondaTigers, alike Anacondas and Tigers used to be wild animals but unlike them, treated so civilized! There are evidence (found in underwater investigations) that they had have excellent communication methods not only by voice, but also via smells, blinks, sudden movements and so on.

Although most of the zoologists believe this creature is extinct, but the followers of this animal's attitude are strong in their faith! They, even, call themselves *Extant AnacondaTigers* and believe the real AnacondaTigers will return before 2050 A.D.

In 1850, Emperor AnacondaTiger (the Wise King) decided to found the **Empire of AnacondaTigers** in the wild island of "AnacondaTigers Island" where all Extant AnacondaTigers were living. He explored entire island and found there were *N* cities in the island, labeled from $\mathbf{1}$ to $\boldsymbol{N}$. Initially all islands were isolated and none of them were connect to any of the others. After consulting with a railroad building company, the company replied him that they can build $\boldsymbol{E}$ possible routes between different cities of the island. Each route is offered as a triple of $< \boldsymbol{u_i}, \boldsymbol{v_i}, \boldsymbol{w_i} >$ which means the company is able to build a bidirectional railroad between cities $\boldsymbol{u_i}$ and $\boldsymbol{v_i}$ in return for getting $\boldsymbol{w_i}$ coins of gold.

Besides, an airline company - who was interested in investing on future of the island and was a true fan of AnacondaTigers! - told the king that as a matter of bounty, they are interested in constructing one (and only one) airline between an arbitrary pair of cities for free (i.e. no cost)!

Now, the king asks you (as a possible fan of AnacondaTiger!) for the cheapest plan of connecting the cities such that there would be at least one path between any two cities $\boldsymbol{u}$ and $\boldsymbol{v}$. A path between cities $\boldsymbol{u}$ and $\boldsymbol{v}$ is defined as a sequence of cities starting from $\boldsymbol{u}$ and ending in $\boldsymbol{v}$ such that any two consecutive cities in the middle are connected, either by a constructed railroad or the exclusive airline.

Input

There are $\mathbf{1} \leq \boldsymbol{T} \leq \mathbf{100}$ tests in the input. This number $\boldsymbol{T}$ is given in first line and then comes $\boldsymbol{T}$ block, each representing one test.

Each test begins with two integers $\mathbf{1} \leq \boldsymbol{N} \leq \mathbf{50,000}$ (number of the cities) and $\mathbf{0} \leq \boldsymbol{E} \leq \mathbf{100,000}$ (number of the offers of the railroad company). Then in the next $\boldsymbol{E}$ lines comes one offer per line with three integers $\boldsymbol{u_i}\ \boldsymbol{v_i}\ \boldsymbol{w_i}$ where $\mathbf{1} \leq \boldsymbol{u_i}, \boldsymbol{v_i} \leq \boldsymbol{N}$ and $\mathbf{0} \leq \boldsymbol{w_i} \leq \mathbf{1,000,000}$. $\boldsymbol{u_i}$ and $\boldsymbol{v_i}$ are distinct for each offer but, yet, there can be more than one offer for any two fixed pair of cities!

Output

For each input test, write the minimum cost of connecting all the cities. If it is not possible for a case, just print **-1** for that.

Sample Input	Sample Output
3 3 3 1 2 1000 2 3 900 3 1 800 4 1 1 2 0 4 3 1 2 100 2 1 100 1 3 1000000	800 -1 1000100

Problem F: Friends For Sale!

Friends For Sale, or shortly FFS, is the name of an online game. In this game, anyone has some cash (amount of money he/she has in $) and a value (how much he/she is worth, again in $).

Initially, a new registered user is worth $\$500$ and has $\$100{,}000$ of cash! The only transaction users can do is to buy each other! When X buys Y, then X becomes the *owner* of Y and Y will be X's pet.

To be more precise, consider a buying action; assume that Z owned Y (had previously bought him) and now X comes and buys Y. Note that X, Y and Z must be necessarily different persons. In this case following steps happen sequentially (assume that before the buy action, Y's value had been $\$W$):

1. X Pays $\boldsymbol{Round}(\mathbf{1.111} \times \$\boldsymbol{W})$ **to** system, where round means nearest integers of a fractional value.
2. Z receives $\boldsymbol{Round}(\mathbf{1.05} \times \$\boldsymbol{W})$ from the system (owner's reimbursement).
3. Y receives $\boldsymbol{Round}(\mathbf{0.05} \times \$\boldsymbol{W})$ from the system (pet's benefit).
4. Y's value (which had been $\$W$) will become $\boldsymbol{Round}(\mathbf{1.1} \times \$\boldsymbol{W})$.

As you see, in each buy, approximately 1.1% of $\$W$ goes back to the system as tax.

Now let's meet with our players! Aideen, Ali and Sahar are 3 addicted persons to this game. Their ids are respectively 1 (Aideen), 2(Ali) and 3(Sahar). In a moment of time, their cash and values are recorded. They have enough cash to raise each other's value but need a 4th person to help them. For instance, assume their values are all $\$100{,}000$ and they own each other (Aideen owns Ali, Ali owns Sahar, Sahar owns Aideen) and their cash are $\$20{,}000$ for Aideen and $\$10{,}000$ for Sahar and Ali. Of course none of them is able to buy both of the other two persons in the same time and they are somehow in a dead end! But, if a 4th person - let's name him Duarte - with enough cash comes and buys Sahar, then Ali can buy Aideen, and after that Sahar buys Ali and finally Aideen takes Sahar back from Duarte! Following table shows the details of all transactions.

Step	Action	Aideen				Ali				Sahar				Duarte	
		value	cash	Pets	owner	value	cash	Pets	owner	value	cash	Pets	owner	Cash	Pets
		1				2				3				4	
	Initially	100,000	20,000	2	3	100,000	10,000	3	1	100,000	10,000	1	2	1,000,000	
1	4 buys 3 =>	100,000	**20,000**	2	3	100,000	**115,000**		1	110,000	**15,000**	1	4	888,900	3
2	2 buys 1 =>	110,000	**25,000**	2	2	100,000	**3,900**	1	1	110,000	**120,000**		4	888,900	3
3	3 buys 2 =>	110,000	**130,000**		2	110,000	**8,900**	1	3	110,000	**8,900**	2	4	888,900	3
4	1 buys 3 =>	110,000	**7,790**	3	2	110,000	**8,900**	1	3	121,000	**14,400**	2	1	1,004,400	

In the above example Duarte has gained $4,400 and cashes of all 3 players are changed. Meanwhile, although in this example Duarte took only one of those 3 at a moment, having enough cash he might buy two (or even all) of them in the same time.

Given Initial condition, which is value, cash and owner of all 3 persons in addition to Duarte's cash, you are asked to help them to boost their values so the minimum value of them be maximum and at the end, they own each other again (no one's owner be Duarte).

You can assume that initially the owner of any of these pets is one of the other two. And they wish to remain so (owned by each other, not Duarte) at the end. It is no problem if one of them own both of the others - just none of them should be a pet of Duarte!

Finally, since they are now good in memorizing the plans, you can assume that after 30^{th} buys action, the game will be over (for a while) and it's not possible to do 31^{st} buy thereafter.

Input

Input consists of multiple test cases. On the first line comes $\mathbf{1} \leq \boldsymbol{T} \leq \mathbf{20}$, the number of test cases.

Each test case comes in 4 lines. Any of first 3 lines describes one person (Aideen, then Ali, then Sahar). In each of these lines 3 integers $\boldsymbol{V_i}\ \boldsymbol{C_i}\ \boldsymbol{O_i}$ comes where V_i is the value of the person, C_i is his/her cash and finally O_i comes that is owner of player i. Finally, in the 4^{th} line of the test case, C_4 comes, that is cash of Duarte, the 4^{th} player.

It's guaranteed that $\mathbf{1} \leq \boldsymbol{O_i} \leq \mathbf{3}$ and $\boldsymbol{O_i} \neq \boldsymbol{i}$.

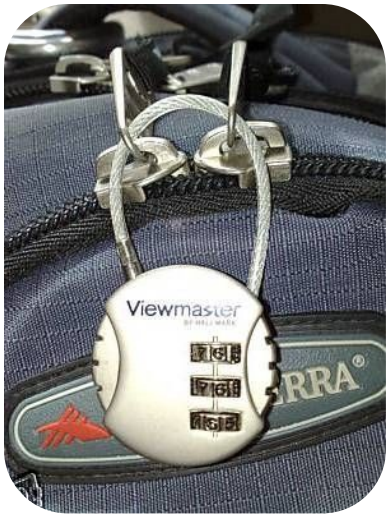
Also, $\mathbf{500} \leq \boldsymbol{V_i} \leq \mathbf{100,000}$ (for first 3 players) and $\mathbf{0} \leq \boldsymbol{C_i} \leq \mathbf{100,000}$ (for all 4 players).

Output

For each test case write the maximum possible "minimum value of the 3 players" after any possible sequence of buys that in the end, 4 has no pet.

Sample Input	Sample Output
2 40145 28600 3 37345 8000 1 40144 6730 2 90000 979 323 2 1077 430 3 1077 350 1 1500	86056 1909

Problem G: Ghandak's Knapsack Key



Ghandak usually hides his most precious things in his knapsack! His knapsack has a usual combination lock, which consists of some dials (each dial consists of the digits 0 to 9). These dials are arranged in two rows. To remember the lock's key, Ghandak memorizes two numbers, the one formed by the digits in the first row, and the one formed by the digits in the second row. Being a mathematician wannabe, Ghandak also memorizes the sum of these two numbers.

Ghandak recently had a car accident, after which he lost parts of his memory. The only things he remembers about the lock are the sum of those two numbers, and the possible digits used in each one.

Before he tries every combination of digits which adhere to his memory, he wants to know how many such combinations are there. And of course you're to help him!

Input

Input consists of multiple test cases. On the first line comes $\mathbf{1} \leq \boldsymbol{T} \leq \mathbf{100}$, the number of test cases.

For each test case first come $\mathbf{1} \leq \boldsymbol{N_1}, \boldsymbol{N_2} \leq \mathbf{18}$ which are the number of dials in the first and second row, respectively. On the next line comes an integer $\mathbf{1} \leq \boldsymbol{S} < \mathbf{10^{18}}$ which is the sum that Ghandak remembers. On the next line come two strings separated by whitespace, consisting of the possible digits used in the first row and the second row. Note that S does not start with zero; but the two numbers in a key may.

Output

For each test case write the number of possible keys that adhere to Ghandak's memory, on a separate line.

Sample Input	Sample Output
1 2 1 102 789 0123465	3

Description of sample output

There are three possible keys which adhere to Ghandak's memory. These are $(\mathbf{97}, \mathbf{5})$, $(\mathbf{98}, \mathbf{4})$ and $(\mathbf{99}, \mathbf{3})$; where the first number in each pair is the one formed by digit selectors in the first row, and second one is the number formed by digit selectors in the second row.

Problem H: How to Make a Robot Dance

A robot is inside a 2-dimensional maze. The maze is a rectangular table of cells. Each cell is filled with either an obstacle or a single decimal digit. In each second, the robot can move to one of the four neighboring cells (if they exist and are not filled with obstacles).

The robot has a small CPU which works with an 8-bit accumulator. Initially the contents of the accumulator are 0. When the robot moves he should choose exactly one operation from *addition*, *subtraction*, *multiplication*, and apply that operation to the accumulator as the first operand, and the digit in the new cell as the second operand. The result is put back into the accumulator.

Note that his accumulator stores only 8-bits of data. So the operations are always done modulo 256. For some strange reason the robot wants to put a determined value in his accumulator.

You're to find him the shortest path to reach that value. If there are many shortest paths, find the one with the lexicographically smallest output string (see output description).

Input

Input consists of multiple test cases. On the first line comes $\mathbf{1} \leq \boldsymbol{T} \leq \mathbf{100}$, the number of test cases.

For each test case first come $\mathbf{1} \leq \boldsymbol{R}, \boldsymbol{C} \leq \mathbf{100}$, the number of rows and columns in the maze. On the next R lines comes the description of the table. An obstacle is represented by `*', and a decimal digit is represented by itself! On the next line come three integers $\mathbf{1} \leq \boldsymbol{r} \leq \boldsymbol{R}, \mathbf{1} \leq \boldsymbol{c} \leq \boldsymbol{C}, \mathbf{0} \leq \boldsymbol{value} \leq \mathbf{255}$ showing the initial row and column of the robot and the value he's trying to put in his accumulator.

Output

For each test case write a single line which describes the choices of the robot. For each move write one of the letters `N', `S', `W', `E' (standing for North, South, West, East) showing the direction of movement, followed by one of letters `*', `+', `-' showing the arithmetic operation done.

If there exists no path to reach the demanded value, print "`No Solution`" instead of entire path.

Sample Input	Sample Output
1 2 5 11011 **1** 2 3 3	N+E+E+

Description of sample output

Initially the robot's accumulator is filled with 1. He moves toward North and adds 0 to his accumulator. Then he moves two steps toward East, adding two 1's to his accumulator.

Problem I: iCalculate!

ACM (Art of Coding with Mathematics) company has produced a new programming language called "+ Calculator 1". This language is a big calculator that supports +, -, *, / and ^ operators and integer numbers.

A code is a prefix expression that satisfies the following conditions:

1- If there is "/ x y" which means ($\frac{x}{y}$), y should be non-zero and x should be divisible by y.
2- If there is "^ x y" which means (x^y), y should be a constant integer (cannot be an expression). So "^ 2 3" is a valid code but "^ 2 + 3 0" is not.
3- All the numbers in the code should be non-negative integers less than 2^{31}.

The language is so simple. A valid code writes the result of the expression on the screen.

ACM company is not able to write their language so it asked you to write a program to compute the result of a valid "+ Calculator 1" code.

Input

First line of input contains an integer T, the number of test cases. Each of the next T lines contains the description of a valid code. Input contains less than $\mathbf{6000}$ characters in total.

Output

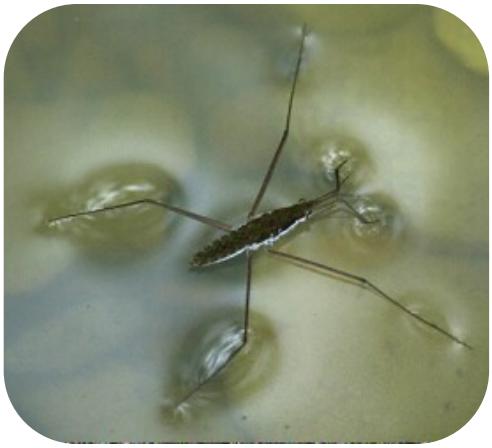
For each test case output a number being the result of that expression. You can assume that all results are integers less than 10^4 by their absolute value.

Sample Input	Sample Output
3 ^ 2 3 + / 4 2 * 3 6 - ^ 2 10 ^ 2 9	8 20 512

Description of sample output

$2^3 = 8, 4/2 + 3 \times 6 = 20, 2^{10} - 2^9 = 512.$

Problem J: Juliet the Water Strider



Juliet the water strider is lost in the middle of Lake Urmia. Her only hope is to reach the nearest shore as soon as possible. But there is a problem; there are some leaves in her way to the shore. She is terribly slow on the leaves. But since she's afraid of losing the shore, she decides to go straightly to the shore and not to change directions.

For this problem let's assume that the lake is the Cartesian plane and Juliet is a point on this plane. Juliet wants to reach the origin of the plane, the nearest point on the shore. Leaves always have either a circular shape or a rectangular one. Note that leaves may overlap with each other.

Your task is to find out the time it takes Juliet to reach the shore.

Input

Input consists of multiple test cases. On the first line comes $\mathbf{1} \leq \boldsymbol{T} \leq \mathbf{100}$, the number of test cases.

For each test case, first come five integers $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{N}, \boldsymbol{v}, \boldsymbol{w}$ where $\mathbf{1} \leq \boldsymbol{N} \leq \mathbf{10000}$ is the number of leaves, $\boldsymbol{v}$ is Juliet's speed on water, $\boldsymbol{w}$ is her speed on leaves, and $(\boldsymbol{x}, \boldsymbol{y})$ is the location of Juliet. On the next $\boldsymbol{N}$ lines, on each line comes the description of one leaf. If the leaf is circular it's described by the string `Circle' followed by the coordinates of the center, followed by the radius (in the same order). If it's rectangular it's described by the string `Rectangle' followed by the coordinates of two opposite corners of the rectangle (the rectangle's sides are always parallel to the axes).

All input numbers are integers.

Output

You should output the time it takes Juliet to reach the shore, rounded down to two decimal digits after the decimal point. Write the result of each test case on a separate line and in the same order as the test cases appear in the input.

Note that you should write each number with exactly two decimal digits after the decimal point.

Sample Input	Sample Output
1 100 0 2 1000 1 Circle 0 0 20 Rectangle 10 10 30 -10	30.07

Description of sample output

Juliet is at the point $(\mathbf{100}, \mathbf{0})$ and wants to reach the point $(\mathbf{0}, \mathbf{0})$. There are two leaves in the lake. One is a circle-shaped leaf centered at $(\mathbf{0}, \mathbf{0})$ with radius $\mathbf{20}$. The other one is a rectangle-shaped leaf with the two opposite corners $(\mathbf{10}, \mathbf{10})$ and $(\mathbf{30}, -\mathbf{10})$. Juliet strides to $(\mathbf{30}, \mathbf{0})$ on the water with a speed of $\mathbf{1000}$. Then he walks on the leaves to the point $(\mathbf{0}, \mathbf{0})$ with a speed of $\mathbf{1}$. Therefore it takes $\frac{\mathbf{100-30}}{\mathbf{1000}} + \frac{\mathbf{30-0}}{\mathbf{1}}$ seconds for her to reach the shore.