# Problem A. Binary Duel

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 20 seconds |
| Memory limit: | 512 mebibytes |

Shark William went to a casino and started playing an unusual game called Binary Duel. The rules are as follows: two players sit at opposite sides of the table, marked as 0 and 1, and then several rounds follow. In each round, both players secretly choose one bit. When both are ready, the bits are revealed and the winner is the player who sits at the side of the table marked with the exclusive-or (xor) of these bits.

While Shark William was playing, his friend, Shark Thomas, did some research and found out that the casino uses androids to play Binary Duel against customers. The androids have perfect memory, and their program is based on the principle "history repeats itself". Formally, if the player's move sequence is $M_1 M_2 \ldots M_n$, the android is looking for the longest suffix of this sequence that is also a substring of the sequence $M_1 \ldots M_{n-1}$, and it assumes that the player will do exactly the same move as he did last time. Knowing that, one can trick the android by simply doing the opposite move of the one predicted by the android's algorithm.

You are given the sequence of moves of Shark William played before both Sharks recognized the android's behaviour. You are to output the sequence of optimal moves for Shark William in several following rounds.

## Input

The first line contains the number of test cases.

Each testcase contains one line with the sequence of bits describing all moves of Shark William in the game before he recognized the android. Then, two integers $n$ and $k$ follow ($1 \leq n, k,\ n + k \leq 10^6$). The length of the sequence is between 1 and 1000 inclusively.

## Output

For each test case, print one line with the sequence of moves of Shark William in $k$ sequential rounds, starting from $n$-th round of the game. The rounds in the game are numbered starting from 1.

## Example

| standard input | standard output |
|---|---|
| 3<br>0 1 10<br>111 10 20<br>10110111 1 20 | 0100110101<br>01010001101011000010<br>10110111001010001100 |

# Problem B. Breeding Sharks

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 3 seconds |
| Memory limit: | 512 mebibytes |

Children of all ages would love to keep sharks as pets (they are so much more interesting than cats or dogs!). Before this can happen, however, geneticists must figure out a way to make sharks less aggressive. A team of researchers has no idea how to do this, but they hope that comparing the genetic codes of various sharks will point them in the right direction.

Specifically, when comparing two DNA sequences ($X$ and $Y$) of equal length, they are looking for a contiguous substring of $X$ such that its mirror reflection is equal to the corresponding substring of $Y$ (the one with the same starting and ending position). The team wants to know the length of the largest of such substrings.

DNA sequences can be awfully long, so you have agreed to find a way to perform the search quickly.

## Input

The first line contains the number of test cases $Z$. The next lines contain $Z$ test case descriptions in the following format.

The first line of each test case contains the number $n$: the length of both DNA sequences ($1 \leq n \leq 10^6$). The second and third lines of each test case contain two sequences of length $n$ consisting of letters '`A`', '`C`', '`G`' and '`T`', one sequence per line. These are the sequences $X$ and $Y$.

## Output

For each test case. write the length of the longest contiguous substring of the first sequence such that its mirror reflection is equal to the substring of the second sequence with the same start and end position.

## Example

| standard input | standard output |
|---|---|
| 1<br>4<br>ACGT<br>TGCT | 2 |

## Note

The substring of "`ACGT`" starting at position 1 and ending at 2 is "`CG`". The substring of "`TGCT`" starting at position 1 and ending at 2 is "`GC`". "`GC`" is equal to the mirror reflection of "`CG`".

# Problem C. Game

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 3 seconds |
| Memory limit: | 512 mebibytes |

Shark Steve wants to self-improve. He has written some pairs of integers on a piece of paper. Now, for each pair $(R, S)$, he wants to know the minimal pair of positive integers $(K_1, K_2)$ (in lexicographic order) such that $K_1 \leq K_2$ and both $R$ and $S$ occur within sequence $(K_n)$ satisfying $K_{i+2} = K_{i+1} + K_i$. The sequence starts from $K_1$.

## Input

The first line contains the number of test cases.

Each test case is described by one line consisting of two integers $R$ and $S$ $(1 < R < S < 10^{18})$.

## Output

For each test case, write one line containing two integers: $K_1$ and $K_2$.

## Example

| standard input | standard output |
|---|---|
| 1<br>8 34 | 1 1 |

# Problem D. Glass

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 10 seconds |
| Memory limit: | 512 mebibytes |

Recently, round windows have become fashionable. Because of that, the service company Cheap Unchipped Glass is introducing a new service to their offer. The factory delivers round pieces of glass of various sizes, and the clients may order them trimmed to smaller circles. Sometimes multiple small circles can be cut out of a single big piece.

Write a program that will optimize glass consumption. Given three glass circles, calculate the minimal radius of the circle which would allow you to cut all three out of it.

## Input

The first line contains the number of test cases.

Each test case consists of three real numbers: the radii of the three glass circles that must be cut out ($0 < r_1, r_2, r_3 \leq 100$).

## Output

For each test case, print one real number on a separate line: the radius of the minimal single piece such that all three given pieces could be cut out of this piece. Output the numbers with absolute error of at most 0.001.

## Example

| standard input | standard output |
|---|---|
| 2<br>1 2 3<br>1 1 0.6666667 | 5<br>2 |

# Problem E. Hunt

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 50 seconds |
| Memory limit: | 512 mebibytes |

Shark Steve has gone hunting. He noticed some fishes moving along straight lines with constant speed. He wants to catch them all. Calculate how fast he could do it. Shark Steve has a given maximum speed. He can change his speed and direction instantly by any amount at any time, as long as he doesn't exceed the maximum speed.

## Input

The first line contains the number of test cases.

The first line of each test case contains three real numbers $x$, $y$ and $v$ ($-100 \le x, y \le 100$, $0 < v \le 100$). Steve is starting at position $(x, y)$ and $v$ is his maximum speed. The second line contains a single integer $n$: the number of fishes to catch ($1 \le n \le 10$). The next $n$ lines describe the fishes. Each fish is described by a single line containing four real numbers $p_x$, $p_y$, $v_x$ and $v_y$ ($-100 \le p_x, p_y \le 100$, $0 < v_x^2 + v_y^2 < v^2$). These numbers mean that the fish starts at $(p_x, p_y)$ and its constant speed vector is $(v_x, v_y)$.

## Output

For each test case, write one line containing a real number: the minimal amount of time Steve needs to catch all fishes. The output will be considered correct if the absolute error is less than $10^{-3}$ or the relative error is less than $10^{-13}$.

## Example

| standard input | standard output |
|---|---|
| 1<br>0 0 2<br>1<br>1 2 -1 0 | 1 |

# Problem F. Knights

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 5 seconds |
| Memory limit: | 512 mebibytes |

Shark Steve is playing a game of Knights. This game is played on a chessboard which initially contains some pawns. The player has to put knights on the chessboard. After the placement, each pawn must be attacked by at least one knight Note that a knight **can** share a square with a pawn. A knight attacks up to eight squares according to the usual rules for a chess knight. A knight does not attack the square it occupies. The chessboard squares are numbered in the following way: $a1$ is given the number 0, $b1 - 1$, $c1 - 2$... finally, the square $h8$ has the number 63. Putting the knight on $i$-th square costs $2^i$ coins. Find the cheapest way to arrange the knights.

## Input

The first line contains the number of test cases.

The first line of a test case contains the number of pawns $n$ ($0 \leq n \leq 64$). The next line contains the positions of pawns (in the chess format: <column letter><row number>).

## Output

For each test case, write a single line containing one integer: the minimal cost of knights' arrangement.

## Example

| standard input | standard output |
|---|---|
| 1<br>6<br>b4 g2 f6 e5 c5 d3 | 268959760 |

# Problem G. Necklace

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 7 seconds |
| Memory limit: | 512 mebibytes |

Shark Steve wanted to impress his girlfriend Gina. He bought a beautiful round necklace with every two neighbouring beads having different colors. When he returned home, he noticed that the necklace does not have this property anymore. It turned out that each bead has two sides (with, possibly, two different colors). The beads can rotate freely — this is how the uniqueness property was lost. Given the necklace's description, help Steve restore its beauty.

## Input

The first line contains the number of test cases.

The first line of each test case contains the number of beads $n$ ($2 \leq n \leq 10^6$). Each of the next $n$ lines contains a pair of uppercase English letters. Each such pair denote the colors of both sides of a bead.

## Output

For each test case, write one line. If the beads of the necklace can be rotated to satisfy the required property (no two consecutive beads having the same color), print a string of uppercase English letters: the chosen colors of each bead ($i$-th character denotes a color present on the $i$-th bead). Note that only one side of the necklace (the one you output) must satisfy the required property. If no solution exists, print a single minus sign ('-').

## Example

| standard input | standard output |
|---|---|
| 1<br>3<br>A B<br>B C<br>C A | ABC |

# Problem H. Of Sharks and Men

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 20 seconds |
| Memory limit: | 512 mebibytes |

Unfortunately, sharks have eaten almost half of the story behind this problem. Below is what remained:

A skilled card dealer in a casino knows the following technique: he takes two decks $A$ and $B$ and shuffles them together, forming a new deck $C$. In the resulting deck, the order of elements originally belonging to $A$ is always preserved, and the order of original elements of $B$ is preserved as well. The dealer is very skilled, so he can produce any deck $C$ he wants, as long as it satisfies the constraints above. After that, the deck $C$ can be further shuffled with other decks.

Every card has a number on it: either 0 or 1. If you buy a deck of cards at a shop, the order of the cards will be always alternating (either `010101010...` or `101010101...`). The dealer wants to produce a deck starting with some given sequence, by shuffling together some number of newly bought decks. The dealer wants to minimize the number of decks used, assuming the number of cards in each new deck is large enough.

To make the task more interesting, you should also determine the smallest $L$ such that decks of length $L$ suffice to produce the desired sequence (while still using the same minimal number of decks).

## Input

The first line contains the number of test cases.

Each test case consists of a single line containing a non-empty sequence of bits of length not greater than $10^6$, describing the sequence the dealer wants to produce.

## Output

For each test case, print two integers: $T$ and $L$. The number $T$ must be the smallest number of new decks needed to produce the desired sequence. The number $L$ must be the smallest deck length that would still allow the dealer to achieve the same goal using only $T$ decks.

## Example

| standard input | standard output |
|---|---|
| 5<br>1<br>11<br>1010<br>1100<br>110110 | 1 1<br>2 1<br>1 4<br>2 2<br>3 3 |

# Problem I. Scaring

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 8 seconds |
| Memory limit: | 512 mebibytes |

Shark Steve is planning to scare some people. By carefully observing the habits of beach-goers, he has learned the moments of time when the beach opens and closes, and also how many people he will scare if he attacks the beach within this time. Attacking a beach takes one unit of time, and he has to start and finish his attack within the opening and closing times of the beach. Unfortunately, after his attack, all the people on the beach run away and the beach is closed, so he can not attack the same beach twice.

Luckily, he is a very fast swimmer, so after he is done attacking one beach, he can instantly swim to another one and start wreaking havoc there. Help him maximize the total number of people scared.

## Input

The first line contains the number of test cases.

First line of test case contains number of beaches $n$ ($0 \le n \le 5000$). Each of the next $n$ lines contains the descriptions of the beaches: three integers $l_i$, $r_i$ and $c_i$ ($-10^9 \le l_i < r_i \le 10^9$, $0 \le c_i \le 10^5$) meaning that he will scare $c_i$ people if he attacks beach number $i$ between moments $l_i$ (its opening time) and $r_i$ (its closing time).

## Output

For each test case, write one line with the maximum number of people shark Steve can scare.

## Example

| standard input | standard output |
|---|---|
| 2<br>2<br>1 3 3<br>1 2 5<br>1<br>1 2 5 | 8<br>5 |

# Problem J. Segmentworm

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 10 seconds |
| Memory limit: | 512 mebibytes |

Shark Brian went to the pond to hang out with other sharks, but on his way, he noticed a little creature he has never seen before. It looked like an earthworm, however, it was built of segments having little legs. Each segment had some number of legs (possibly zero). Brian was fascinated by this little being, so he went back to his home and did some research on the Web. It turned out that this little being is called a *Segmentworm* and has a very interesting growing pattern. Generally, a Segmentworm can be imagined as a chain of segments. When a Segmentworm is born, it has only one segment with no legs. Every day, each segment can either grow a new leg or split into two segments, or remain in their current state. Every day, there is at least one segment that does one of these two things. Also, during the splitting process, legs are distributed arbitrarily between the resulting two segments (with possibly all legs going to only one of them). Now, Shark Brian wonders what is the minimal age (in days) of the Segmentworm he has seen at the pond. Help him!

## Input

The first line contains the number of test cases.

Each test case describes one Segmentworm. Each description contains the number $n$ ($1 \leq n \leq 5 \cdot 10^6$) followed by $n$ numbers $a_i$ ($0 \leq a_i \leq 10^6$). These numbers mean that the Segmentworm had $n$ segments, and for each $i$, segment number $i$ had $a_i$ legs.

## Output

For each test case, print one integer on a separate line: the minimal age of the Segmentworm.

## Example

| standard input | standard output |
|---|---|
| 3<br>1 0<br>1 5<br>4 0 0 0 0 | 0<br>5<br>2 |

# Problem K. Watchshark

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 20 seconds |
| Memory limit: | 512 mebibytes |

A group of sharks is preparing a massive robbery targeting the Coral Reef Bank. Shark Tony is responsible for reconnaissance. It is his job to observe all the points which are critical for the plan to succeed. This task is very difficult because of the large distances separating those points.

To address this issue, the group is considering hiring another shark to help Tony. However, this would introduce additional risk and the new shark would have to get a share in the profits! For this reason, the sharks want to know in advance whether adding a new member to their team would really help Tony in his task.

You are given the coordinates of each of the important points. Your task is to decide if they can be split into two sets so that each has smaller diameter than the original set. The diameter of a set is the largest distance between any two points in the set.

## Input

The first line contains the number of test cases $Z$. The next lines contain $Z$ test case descriptions in the following format.

The first line of each test case contains the number $N$: the number of important points ($1 \leq N \leq 10^5$). Each of the next $N$ lines contains three integers $x_i$, $y_i$ and $z_i$: the coordinates of the $i$-th point. Each coordinate's absolute value is not greater than $8 \cdot 10^8$. All points are pairwise distinct and they all lie on the plane defined by the equation $x + y + z = 0$.

## Output

For each test case, print one line with a single word in it. If the given set of points can be split into two sets so that each has a smaller diameter than the original, output the word "`TAK`". Otherwise, output the word "`NIE`".

## Example

<table><tr><th>standard input</th><th>standard output</th></tr><tr><td><pre>3
1
1 1 -2
2
1 -1 0
-1 1 0
3
1 -1 0
0 1 -1
-1 0 1</pre></td><td><pre>NIE
TAK
NIE</pre></td></tr></table>