# Problem A. The Catcher in the Rye

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 1 second |
| Memory limit: | 512 mebibytes |

Being an experienced jogger, you decided to take a shortcut through a rectangular grain field, divided into three vertical patches full of rye, wheat and barley, respectively. You want to travel from its bottom left corner to the top right corner.

The proper jogging technique varies depending on which sort of grain you are trodding. Therefore, you will be going through the patches with three possibly different velocities. You may plan your route any way you wish (but without leaving the field).

Find the minimal time needed to get to the top right corner of the field from the bottom left corner.

## Input

The first line of input contains the number of test cases $z$ ($1 \le z \le 10\,000$). The descriptions of the test cases follow.

Each test case is given on a single line containing seven integers $h, a, b, c, v_a, v_b, v_c$ ($1 \le h, a, b, c, v_a, v_b, v_c \le 10^5$): the height of the rectangular field, the widths of the patches, and the allowed velocities in these patches. Velocity $v$ means that you travel exactly $v$ units of distance per time unit.

## Output

For each test case, output the total time needed to get from the bottom left corner to the top right one.

Your answer will be accepted if it differs from the correct one by no more than $10^{-3}$.

## Example

| standard input | standard output |
|---|---|
| 2<br>10 3 4 3 1 1 1<br>21 5 12 4 4 3 4 | 14.14213562<br>8.75 |

# Problem B. Dissertation

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 1 second |
| Memory limit: | 512 mebibytes |

A professor suspects that a student's dissertation was plagiarized from a certain book. In order to test that, he wants to compute the longest common subsequence of the dissertation and the book. He doesn't have a program to do it, so he asked you to write such a program as an assignment for the algorithms course.

## Input

The first line of input contains the number of test cases $z$ ($1 \leq z \leq 10^9$). The descriptions of the test cases follow.

Each test case is given on two lines. The first line contains a string of length between 1 and 1 000 000 consisting of lowercase Latin letters: the text of the book. The second line contains a string of length between 1 and 1000 consisting of lowercase Latin letters: the text of the dissertation.

The sum of lengths of the books in all test cases is at most 10 000 000. The sum of lengths of the dissertations in all test cases is at most 30 000.

## Output

For each test case, output the length of the longest common subsequence of the book and the dissertation.

## Example

| standard input | standard output |
|---|---|
| 1<br>abcdefghijklmnopqrstuvwxyz<br>bbddee | 3 |

# Problem C. Dominoes

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 1 second |
| Memory limit: | 512 mebibytes |

One rainy day Lech and Grzegorz were killing time playing with dominoes. Their domino set is a bit larger than ordinary ones: it contains $\frac{n(n+1)}{2}$ pieces, each of them consisting of two halves with a single integer. The pieces are numbered (1 1), (1 2), ..., (1 n), (2 2), (2 3), ..., (n n).

As the players quickly got bored of following regular rules, they started thinking about something more challenging. Grzegorz came up with the following idea: put all the pieces together so that for every integer, half-pieces with that integer would occupy a connected area (all 1s together, all 2s together, and so on). They started putting the pieces, but it got very hard, very fast. "I know what we should do!", Lech laughed devilishly, "Let's give this task to our students!".

You should place all pieces on a large board divided into unit squares. Each domino should occupy exactly two unit squares, each containing one number. It may be placed horizontally or vertically. Pieces cannot overlap. Two unit squares are considered connected if they share an edge.

## Input

The first line of input contains the number of test cases $z$ ($1 \le z \le 50$). The descriptions of the test cases follow.

Each test case is given on a single line containing a single number $n$ ($1 \le n \le 1000$).

## Output

For each test case, if there is no solution, output "`impossible`" on a separate line. Otherwise, output $\frac{n(n+1)}{2}$ lines containing positions of the pieces, starting with (1 1), (1 2), ..., and ending with the position of the (n n)-piece. Each position should be presented as four integers $x_1, y_1, x_2, y_2$ ($0 \le x_1, x_2, y_1, y_2 \le 10\,000\,000$), where $x_1, y_1$ are the coordinates of the non-greater number on the piece and $x_2, y_2$ are the coordinates of the non-smaller number.

## Example

| standard input | standard output |
|---|---|
| 1<br>2 | 1 1 1 2<br>1 3 1 4<br>1 5 1 6 |

# Problem D. Endgame

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

The game of chess is almost finished. On the chessboard, apart from White and Black kings, there is only a White rook.

You are playing White, and it is your move. Determine the minimal number of moves you need to give a checkmate, provided that your opponent plays optimally and delays his inevitable defeat for as long as possible.

There is a compilation of chess rules at the end of this statement. If you already know them, rest assured: your puny chess skills will not help you solve this problem.

## Input

The first line of input contains the number of test cases $z$ ($1 \leq z \leq 10$). The descriptions of the test cases follow.

Each test case is given on eight lines describing a chessboard. Each of these lines describes a single row and contains exactly eight characters: '`.`' denotes an empty field, '`W`' is the White king, '`B`' is the Black king, and '`R`' is the White rook. There is exactly one piece of each kind. The starting position is guaranteed to be valid: in particular, kings are not adjacent to each other, and the Black king is not under attack.

There is an empty line after each test case.

## Output

For each test case, output a line containing a single integer: the maximal possible number of moves White needs to give a checkmate (per common tradition, count only your moves, not Black's).

## Example

| standard input | standard output |
|---|---|
| 2<br>........<br>........<br>........<br>........<br>........<br>.......W<br>R.......<br>.......B<br><br>....B...<br>........<br>..W.....<br>........<br>.....R..<br>........<br>........<br>........ | 1<br>2 |

## Note

Chess rules:

- The players alternately move one piece per turn.
- A player cannot "pass"; on each turn, they have to make a legal move.
- The king moves one square in any direction (horizontally, vertically, or diagonally).
- The rook can move any number of squares along any row or column, but may not leap over other pieces.
- A king is *under attack* if it is within move range of an opposing piece.
- A player may not make any move that would put or leave his or her king under attack (in particular, the king cannot be moved to a square adjacent to other king).
- A Black king can, however, move to a square occupied by the White rook, if the White king is not adjacent to the rook. The rook is then captured and the game ends in a draw.
- If Black player has no legal move, the game is over; it is either a checkmate (White wins) if the Black king is under attack, or a stalemate (a draw) if it is not.
- It is known that, in the situation described above (king and rook vs. king), a checkmate is always possible in less than 50 moves.

# Problem E. Evacuation

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 4 seconds |
| Memory limit: | 512 mebibytes |

It is a dark and stormy night. Our agent has just performed a dangerous mission in Central America. His life is in danger, so he must be evacuated at once. The only place from which a helicopter can pick him up is a straight road going through the jungle. The agent starts at the middle of the road (marked as position 0) and can walk in either direction with the speed of 1 meter per second (it is, as we said before, a dark and stormy night).

Unfortunately, weather conditions have just gotten even worse. A thunderstorm is coming our way, with its lightning strikes being a danger to both the helicopter and the agent. Therefore, there are only a few set times and places at which our helicopter can land. With our state-of-the-art equipment, we can predict where and when exactly will the lightning strike, and the area that is going to be burned. With this knowledge, determine which landing spots the agent can reach on time, avoiding the strikes.

Note that if both landing spot and lightning strike have the same time $t$ and they are closer to each other than $r$, the evacuation is impossible

## Input

The first line of input contains the number of test cases $z$. The descriptions of the test cases follow.

The first line of each test case contains an integer $n$ ($0 \le n \le 200\,000$): the number of lightning strikes. Each of the following $n$ lines describes an expected lightning strike, containing three integers $t, x, r$ ($0 \le t \le 10^9$, $-10^9 \le x \le 10^9$, $1 \le r \le 10^9$): the time $t$, position $x$ and blast radius $r$ of the strike. The agent should be at least $r$ meters away from the lightning at time $t$, or he will be fried.

The next line contains the number of possible landing spots $s$ ($1 \le s \le 200\,000$). Each the following $s$ lines contains two integers $t, x$ ($0 \le t \le 10^9$, $-10^9 \le x \le 10^9$) meaning that the helicopter can land at time $t$ and position $x$. You must determine if the agent can also be there at this time.

The total number of all lightning strikes and landing spots together in all test cases does not exceed 1 300 000.

## Output

For each test case, determine for all landing spots if the agent can be there at given time (without being struck by lightning), and output a line containing the answers. For each landing spot in the order they are given, output a single character '`@`' if the agent can be evacuated from that spot, and '`*`' otherwise.

## Example

| standard input | standard output |
|---|---|
| 1<br>2<br>3 -1 1<br>2 1 1<br>3<br>1 1<br>3 1<br>2 1 | @@* |

# Problem F. Factory

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

"ASD Inc." is a worldwide quadcopter manufacturer. They have recently decided to build a new factory, which will make a brand new line of quadcopters. However, the company has not yet decided where to build it.

The factory should be placed in such location that the cost of delivering produced quadcopters to stores is minimal possible. There are $n$ stores that are interested in selling the new model. The produced quadcopters will deliver themselves to the stores, each one flying straight from the factory to its selected store. If the Euclidean distance between the factory and some store equals $x$, then the monthly cost of delivery is exactly $x$ bitcoins.

The factory can be built anywhere, even if there is some store in this point (in that case, delivery is costless). Given the coordinates of the stores find the optimal place to build the factory.

## Input

The first line of input contains the number of test cases $z$ ($1 \leq z \leq 10$). The descriptions of the test cases follow.

The first line of each test case contains the number of stores $n$ ($2 \leq n \leq 1000$).

Each of the next $n$ lines contains two integers $x, y$ ($-10^6 \leq x, y \leq 10^6$) denoting coordinates of the stores.

You may assume that no two stores occupy the same spot.

## Output

For each test case, output two numbers denoting the coordinates of a point which minimizes sum of distances to the stores. If there are multiple such points, output any one of them.

If the cost of the optimal solution is $x$, then solution with cost $y$ will be considered correct if $|\frac{y-x}{x}| < 10^{-6}$.

## Example

| standard input | standard output |
|---|---|
| 1<br>3<br>-3 0<br>0 3<br>3 0 | 0.000000 1.732051 |

# Problem G. Grasshoppers

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 15 seconds |
| Memory limit: | 512 mebibytes |

While relaxing on a meadow you've noticed an incredible spectacle: a group of grasshoppers was jumping in a circle. You found the dance particularly beautiful, as you realized that their moves were not random, but followed a mathematical pattern.

There are $m$ points marked on the circle. These points are numbered from 1 to $m$ in the order they appear on the circle and divide the circle into arcs of equal length. There are grasshoppers in some of these points, possibly more than one grasshopper in the same point. The grasshoppers are numbered from 1 to $n$. Each second grasshoppers jump to new locations according to the following rule: If at the beginning of the second the grasshoppers $1, 2, \ldots, n$ are standing in points $A_1, A_2, \ldots, A_n$ respectively, and $O$ is the center of the circle, then at the end of the second the grasshoppers will be standing at positions $B_1, B_2, \ldots, B_n$, where $B_k$ is the reflection of point $A_k$ over the line $OA_{k+1}$ for $k = 1, 2, \ldots, n-1$, and $B_n$ is the reflection of point $A_n$ over the line $OA_1$. The grasshoppers' numbers do not necessarily correspond to their order in the circle, and do not change during the dance.

You need to go back home now, but you are wondering what will happen later on. Given the initial arrangement of the grasshoppers, find their positions after $t$ seconds.

## Input

The first line of input contains the number of test cases $z$ ($1 \le z \le 10^9$). The descriptions of the test cases follow.

The first line of each test case contains three integers $n, m, t$ ($1 \le n \le 100\,000$, $3 \le m \le 100$, $1 \le t \le 10^9$): the number of grasshoppers, the number of arcs and the number of seconds. The second line contains $n$ integers denoting initial positions of the grasshoppers. The positions are integers between 1 and $m$ inclusive. The total number of grasshoppers in all test cases does not exceed 200 000.

## Output

For each test case, output positions of the grasshoppers after $t$ seconds, separated by spaces.

## Example

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

# Problem H. Education Nightmare

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

You are having a nightmare! In the dream, you have finished your education and you are now to start teaching others. And today is your first day! You enter the school building (being a little late) and... of course, you have forgotten which room your class takes place in.

The building consists of $n$ rooms, numbered $1, 2, \ldots, n$. Some pairs of rooms are connected with a passage. You may move between connected rooms, each transfer taking exactly one second. There are exactly $n - 1$ passages in the building, and every room can be reached, given enough time. In other words, the rooms and passages form a **tree**.

You start in room $s$. If you enter the correct room, you will immediately recognize it and end your quest. Even so, searching all the rooms could take really long... Fortunately, there is one more trick you can use: in the room $m$ there is a complete timetable of all classes. If you ever enter room $m$, you immediately learn where the class is, and you may go there at once, using the shortest route (be advised, though: the room $m$ does not provide any printing, faxing, scanning or photocopy services. You shouldn't ask them about it).

Find the minimal time $t$ needed to find your class in the worst case, that is, the minimal number $t$ for which there is a strategy guaranteeing finding the correct room in $t$ seconds.

## Input

The first line of input contains the number of test cases $z$ ($1 \le z \le 10^9$). The descriptions of the test cases follow.

The first line of each test case contains three integers: the number of rooms $n$, the starting room $s$, and the room with the timetable $m$ ($1 \le n \le 200\,000$, $1 \le s, m \le n$). Then, $n-1$ lines follow, each containing two integers $a, b$ ($1 \le a, b \le n$, $a \ne b$), denoting a passage between rooms $a$ and $b$. It is guaranteed that every room can be reached from every other one.

It is possible for the class to take part in room $s$ or in room $m$. It is also possible to start in room $m$.

The total number of rooms in all test cases does not exceed $10^7$.

## Output

For each test case, output a single integer: the total number of seconds needed to reach the classroom, assuming the best possible strategy.

## Example

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

# Problem I. A Really Odd Sequence

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 6 seconds |
| Memory limit: | 512 mebibytes |

According to our long-established tradition, the best statements are those kept short.

Given a sequence of integers, find the largest sum of a consecutive subsequence of odd length.

## Input

The first line of input contains the number of test cases $z$. The descriptions of the test cases follow.

The first line of each test case contains the length of the sequence $n$ ($1 \leq n \leq 1\,000\,000$).

The next line contains $n$ integers $a_1, a_2, \ldots, a_n$ ($-10^9 \leq a_i \leq 10^9$), the elements of the sequence.

The total length of all sequences in all test cases does not exceed $5\,000\,000$.

## Output

For each test case, output the largest sum on a separate line.

## Example

| standard input | standard output |
|---|---|
| 1<br>4<br>8 -7 9 1 | 10 |

# Problem J. Spoonerisms

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 2 seconds |
| Memory limit: | 512 mebibytes |

A *spoonerism* (named after William Archibald Spooner, an Oxford pastor who had a habit of inadvertently inventing more of them) is a pair of words that you can change into another pair by swapping their beginnings, for example a "`blushing crow`" becomes a "`crushing blow`".

Given a list of words, find a spoonerism among them. Formally: find a pair $(A, B)$ of words from the list which can be split into $A = pq$ and $B = rs$ in such a manner that the words $C = rq$ and $D = ps$ are also on the list. We allow only true spoonerisms, that is, those with $p \neq r$, $s \neq q$ and $p, q, r, s$ nonempty.

## Input

The first line of input contains the number of test cases $z$. The descriptions of the test cases follow.

The first line of each test case contains the length of the list $n$ ($1 \leq n \leq 500\,000$). Each of the following $n$ lines contains a single word composed of small English letters. The total length of words in all test cases does not exceed 500 000.

## Output

For each test case, if no spoonerism can be found, output "`NO`" on a single line. If there is a spoonerism, output a line containing "`YES`", followed by a line containing words $A$ and $B$, and another one containing $C$ and $D$. If there are multiple solutions, output any one of them. You may also safely switch the word order in any line.

## Example

| standard input | standard output |
|---|---|
| 1<br>9<br>blunder<br>blushing<br>crow<br>cry<br>crushing<br>blow<br>black<br>back<br>clap | YES<br>blushing crow<br>crushing blow |

# Problem K. A Text Problem

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 6 seconds |
| Memory limit: | 512 mebibytes |

The string $A$ occurs in the string $B$ at position $i$ *with at most one mistake* if and only if either $A$ occurs in $B$ at position $i$, or there exists a string $A'$ obtained from $A$ by replacing the letter at a single position with a different letter such that $A'$ occurs in $B$ at position $i$.

You are given a string $T$ and a series of queries. Each query is a string for which you should compute the number of positions at which it occurs in $T$ with at most one mistake.

## Input

The first line of input contains the number of test cases $z$. The descriptions of the test cases follow.

The first line of each test case contains a string of length between 1 and 200 000 consisting of lowercase Latin letters: the string $T$. The next line contains one integer $q$: the number of queries. Each of the following $q$ lines contains a nonempty string consisting of lowercase Latin letters: a query. The sum of lengths of all queries in a test case is at most 200 000.

The sum of lengths of all strings appearing in all test cases (including queries) does not exceed 1 200 000.

## Output

For each query, output the number of positions in $T$ at which the query occurs with at most one mistake.

## Example

| standard input | standard output |
|---|---|
| 1<br>abcdefghij<br>2<br>abd<br>a | 1<br>10 |

# Problem L. Waiter's Problem

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 1 second |
| Memory limit: | 512 mebibytes |

Cafe Satori hires only experienced waiters, and there is a reason for that. Everyday during lunchtime, at 12:00 noon, a horde of hungry customers enters the cafe and waits to be served. Therefore, the only waiter on duty has his hands full. Fortunately, his heroic work will be rewarded with generous tips.

Each customer is willing to leave a specific amount of tip, provided he is served immediately. For each minute of waiting the tip decreases by one, until it reaches zero.

It takes one minute for the waiter to serve a customer. Solve the waiter's problem and calculate how much money he can get, provided he serves customers in an optimal order.

## Input

The first line of input contains the number of test cases $z$ ($1 \le z \le 10^9$). The descriptions of the test cases follow.

The first line of each test case contains the number of customers $n$ ($1 \le n \le 100\,000$). The second line contains $n$ non-negative integers not exceeding $10^9$: these are the amounts of tips the customers are willing to give initially.

The total number of customers in all test cases does not exceed $1\,000\,000$.

## Output

For each test case, output the maximum total amount of tips the waiter can get.

## Example

| standard input | standard output |
|---|---|
| 1<br>6<br>0 9 9 1 7 5 | 24 |