Preliminaries

for the 2012 Benelux Algorithm Programming Contest

The Problem Set

A	Annoying Mosquitos
B	Bad Scientist
C	Cubing
D	Digital Transmission
E	Extreme Shopping
F	Float to Fraction
G	Gunslinger
H	Hex
I	Islands
J	Jumbled Letters

Almost blank page

A Annoying Mosquitos

Lee wants to go to bed but there are mosquitos on the wall in his room. He knows that they will try to bite him as soon as he is about to fall asleep, as they have been doing for the past couple of days. Since he appreciates the value of a good night sleep very much, he decides enough is enough and gets his fly swatter.

Unfortunately for him he was born with the distinct swatting disadvantage of being completely blind. The insects seem to comprehend this and keep really still to avoid triggering his heightened sense of hearing. Lee has no other choice but to hit the wall randomly, but luckily he's got a pretty large fly swatter: each swat hits all mosquitos in a square area of 101 by 101 units.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with an integer n ($1 \leq n \leq 100$): the number of mosquitos on the wall.
- n distinct lines with two space-separated integers x_i and y_i ($-1\,000 \leq x_i, y_i \leq 1\,000$): the position of the i-th mosquito.
- one line with an integer m ($1 \leq m \leq 10\,000$): the number of swats Lee tries.
- m lines with two space-separated integers x_j and y_j ($-1\,000 \leq x_j, y_j \leq 1\,000$): the midpoint of the j-th attempt.

Output

Per test case:

- one line with the number of mosquitos that get hit.

Sample in- and output

Input	Output
2 3 15 -10 16 40 17 41 1 15 -10 1 100 100 3 90 90 100 110 -500 -400	2 1

Almost blank page

B Bad Scientist

You are a rather fraudulent scientist who makes up all his research data. Due to careful planning you got away with this for many years, but recently you became sloppy. In your most recent paper you came up with so many theories that some even contradict each other. You want to throw away as few theories as possible such that no contradicting pair remains.

There is a problem however: your colleagues have seen you doing a lot of research. There is a limit on how many theories you can throw away before they get suspicious.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with an integer n ($1 \leq n \leq 50$): the number of theories in the paper.
- one line with an integer k ($0 \leq k \leq 16$): the maximum number of theories that can be discarded without arousing suspicion.
- one line with an integer m ($0 \leq m \leq n \times (n-1) / 2$): the number of theory pairs that contradict each other.
- m distinct lines with two space-separated integers x_i and y_i ($1 \leq x_i < y_i \leq n$), indicating theory x_i and theory y_i are contradictory.

Output

Per test case:

- one line with the minimal number of theories that have to be discarded to make a consistent paper or the word "`IMPOSSIBLE`" if this is impossible without arousing suspicion.

Sample in- and output

Input	Output
2 5 5 2 1 3 2 3 3 1 3 1 2 2 3 1 3	1 IMPOSSIBLE

Almost blank page

C Cubing

A Rubik's Cube is a three-dimensional puzzle. In its original version, it is made up of 3×3×3 smaller cubes. Each of the visible faces of these smaller cubes has one of six colors (white, yellow, red, orange, green or blue). A Rubik's Cube is said to be solved if on each face, the nine cubes all show the same color.

The Cube is mechanically constructed such that it is possible to rotate any face (that is, any set of 3×3 smaller cubes making up one of the six faces of the entire cube) by 90 degrees in both directions. When a rotation is completed, it is again possible to rotate any face in any direction. Thus by combining rotations performed on different faces, it is possible to mix up the colors. The puzzle is then to take a Cube whose colors are mixed up, and return it to its solved state.

We start from a solved Rubik's Cube, which is completely white on the top (up) face, yellow on the bottom (down), red on the front, orange on the back, green on the left and blue on the right. You are given a sequence of rotations. Determine what the top face looks like after applying all rotations.

A solved Rubik's Cube. In the left image the top (white), front (red) and left (green) face are shown. In the right image the same cube is viewed from the opposite side, with the bottom (yellow), back (orange) and right (blue) face visible. Shown is the way in which a face can be rotated (in this case, the left face is partially rotated clockwise).

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with an integer n ($1 \leq n \leq 1\,000$): the number of rotations.
- one line with n space-separated pairs of characters: the rotations. For each rotation, the first character identifies the face that is rotated: '`U`' for the top (up) face, '`D`' for the bottom (down), '`F`' for the front, '`B`' for the back, '`L`' for the left and '`R`' for the right. The second character indicates the direction of the rotation: '`+`' for a clockwise rotation (when looking directly at the face that is rotated), '`-`' for a counterclockwise rotation.

Output

Per test case:

- three lines, each with three characters: the colors of the squares on the top face of the cube, after applying the rotations. The first line corresponds to the squares adjacent to

the back side of the cube. The colors are indicated by the first letter (lower case): '`w`' for white, '`y`' for yellow, '`r`' for red, '`o`' for orange, '`g`' for green and '`b`' for blue.

Sample in- and output

Input	Output
4 1 L- 2 F+ B+ 4 U- D- L+ R+ 10 L- U- L+ U- L- U- U- L+ U+ U+	rww rww rww bbb www ggg gwg owr bwb gwo www rww

D Digital Transmission

A line code is an encoding of a digital signal suitable for serial transmission over a physical medium, such as copper wire or optical fiber. Such codes are designed to prevent distortion of the signal due to the physical properties of the transmission medium.

For this problem we consider a line code that outputs a binary string (i.e. a sequence of zeros and ones) with two restrictions. The output must be:

1. *DC-balanced*: it must contain an equal number of zeros and ones;
2. *run-length limited*: it may not contain more than two consecutive zeros or ones.

You have received a transmission and you wonder if it conforms to the above specification. This would be easy to verify, except that you were not able to receive every symbol properly, resulting in blanks (encoded as periods) in the string, which could have been either a zero or a one.

Write a program that determines for a given string whether it is possible to fill in each blank with either zero and one so that the result is a valid line code.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with a string consisting of `'0'`, `'1'` and `'.'`: the received transmission. The length of the string is even and between 2 and 100 000.

Output

Per test case:

- one line with the word "`yes`" or "`no`", indicating whether or not the blanks in the string can be replaced by zeros and ones to create a properly coded string.

Sample in- and output

Input	Output
6 001100110011 110..0...00...0011 .... 01....100100 010101110100 101011011001	yes yes yes no no no

Almost blank page

E Extreme Shopping

John wants to buy a large number of widgets. He has already compiled a list of shops that sell the widgets he likes.

Each shop may charge a different price per widget. Additionally, each shop only has a limited number of widgets in stock (and John does not want to order out-of-stock items). Finally, each shop may charge a fixed per-order fee, that John must pay once, if he decides to place an order at that shop. Of course, a single order can include any number of widgets (up to the amount in stock).

You will be given the pricing information for all shops. Your task is to determine how much money John has to spend to obtain the desired number of widgets if he spreads his orders optimally across the various shops.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with two space-separated integers n ($1 \leq n \leq 10\,000$) and m ($1 \leq m \leq 100$): the number of items to buy and the number of shops available.
- m lines with three space-separated integers s_i ($0 \leq s_i \leq 10\,000$), p_i ($0 \leq p_i \leq 10\,000$) and o_i ($0 \leq o_i \leq 1\,000\,000$): the number of items in stock, the price per item and the price per order, for each shop.

The desired number of items does not exceed the total number of items available in all shops combined (i.e. $n \leq \sum_{i=1}^{m} s_i$).

Output

Per test case:

- one line with an integer: the minimum possible total amount of money John needs to spend to obtain n widgets.

Sample in- and output

Input	Output
2 20 4 5 5 6 10 4 12 15 6 9 20 7 0 10 2 5 0 50 1000 10 0	118 100

Almost blank page

F Float to Fraction

When you write a fraction in decimal notation, the sequence of decimals will either terminate (e.g. $^1/_8 = 0.125$), or at some point a sequence of digits will repeat itself indefinitely (e.g. $^1/_{11} = 0.09090909\ldots$). The reverse is also true: if the sequence of decimals terminates, or a sequence repeats itself, the number can be written as a fraction.

Write a program which, given a number in decimal notation, returns the fraction.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- a decimal number on a single line, given as follows:
  - The first two characters are "`0.`"
  - This is followed by a sequence of digits, of length between 0 and 6.
  - Optionally, a sequence of digits between parentheses follows, of length between 1 and 9. This represents the sequence which repeats itself indefinitely. When this sequence is absent, it means that the sequence of decimals has terminated.

  There will be at least one non-zero digit. The sequence between parentheses (if present) does not consist of zeros only nor of nines only.

Output

Per test case:

- one line with two positive integers, relatively coprime (i.e. they do not share any factors apart from 1), separated by a forward slash ('`/`'), representing the numerator and denominator respectively of the reduced fraction which equals the number given in the input.

Sample in- and output

Input	Output
3 0.5 0.(3) 0.6(142857)	1/2 1/3 43/70

Almost blank page

G Gunslinger

The quickly shooting, maiden saving, gunslinging cowboy Luke finds himself in the den of his archenemy: the Dalton gang. He is trying to escape but is in constant danger of being shot. Fortunately his excellent marksmanship and quick reflexes give him the upper hand in a firefight: the gang members are all too scared to move, let alone draw their guns. That is, as long as Luke can see them. If he cannot see one of the thugs, then that Dalton will immediately fire upon Luke and kill the cowboy without fear of retaliation. Luke's amazing eyesight allows him to cover a field of view of 180 degrees all the time. While doing so he can move around freely, even walking backward if necessary.

Luke's goal is to walk to the escape hatch in the den while turning in such a way that he will not be shot. He does not want to shoot any of the Daltons because that will surely result in a big fire fight. The Daltons all have varying heights, but you may assume that all the people and the hatch are of infinitesimal size.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with two space-separated integers x_L and y_L: Luke's starting position.
- one line with two space-separated integers x_E and y_E: the position of the escape hatch.
- one line with an integer n ($1 \leq n \leq 1\,000$): the number of Dalton gang members.
- n lines with two space-separated integers x_i and y_i: the position of the i-th Dalton.

All x and y are in the range $-10\,000 \leq x, y \leq 10\,000$. Luke, the escape hatch and all Daltons all have distinct positions.

Output

Per test case:

- one line with the length of the shortest path Luke can take to the escape hatch without dying, rounded to three decimal places, or "`IMPOSSIBLE`" if no such path exists.

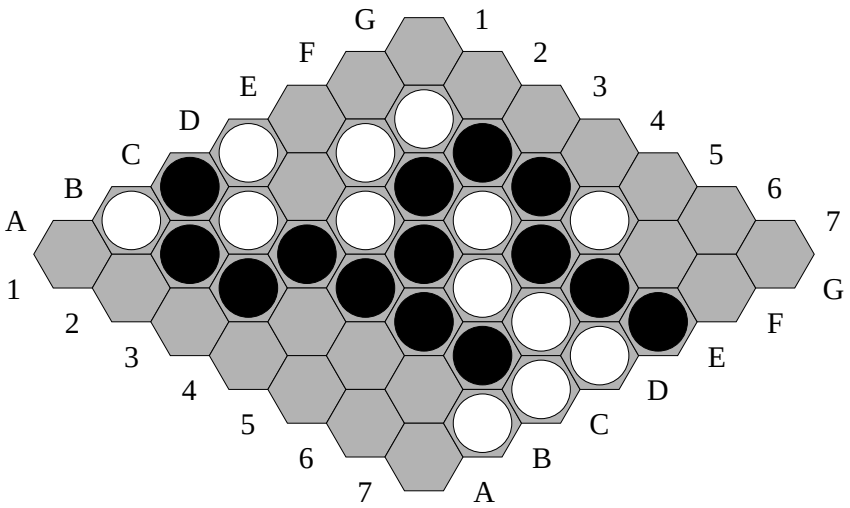
The test cases are such that an absolute error of at most 10^{-6} in the final answer does not influence the result of the rounding.

Sample in- and output

Input	Output
2 0 0 2 0 2 1 1 1 -2 0 0 2 0 3 1 1 1 -2 -1 0	2.828 IMPOSSIBLE

H Hex

Hex is a game for two players, played on a diamond-shaped board with hexagonal cells. At the start of the game, all cells are empty. Each player in turn puts a stone of his own color (black or white) in any empty cell. The goal for Black is to connect the top left edge of the board with the bottom right edge, by making a path consisting of neighboring cells with Black stones in them. White attempts to make a path from the top right to the bottom left. The cells located in the corners of the diamond count as being adjacent to both corresponding edges. The player to start the game is Black. An interesting feature of this game is that there is always a winner: if all cells are filled and one player does not have a path connecting his designated edges, then the other player automatically does have one.



The game situation as described by the first test case in the sample input. Black has successfully connected the top left edge and the bottom right edge and has therefore won the game.

Write a program which, given a game situation, determines which player has won, or that the game has not yet finished.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with one integer n ($2 \leq n \leq 100$): the size of the board (the number of cells along each edge).
- n lines with n characters: the i-th line lists the contents of the i-th diagonal of the board from the bottom left edge to the top right edge (these diagonals are labeled 1-7 in the figure). The diagonal along the top left edge of the board (A1-G1 in the figure) is given first and the diagonal along the bottom right edge (A7-G7) is given last.
  On each line, the first character represents the contents of the cell along the bottom left edge (labeled A in the figure) and the last concerns the cell along the top right edge (G). Each character is '`B`', '`W`' or '`.`', representing a black stone, white stone, or empty cell respectively.

The number of cells with black stones is equal to, or one more than, the number of cells with white stones. The game situation can be such that one of the players had already won a few turns earlier.

Output

Per test case:

- one line with either "`Black wins`" or "`White wins`" or "`Not finished`", indicating the game status.

Sample in- and output

Input	Output
3 7 .WBW... .BW.WW. .BBWBB. ..BBWB. ..BWBW. ..BWB.. .WWWB.. 5 ..B.. .BBWB WWWBW B.WWB .B... 4 BBWB WWB. BWWB BWBW	Black wins White wins Not finished

I Islands

The financial crisis in Greece has major consequences for the Greeks, especially for those who live on one of the many islands. Some of them can not even afford to travel from one island to another by boat. They avoid having to go to another island as much as possible, but if they really must, they have to swim.

Since swimming is very exhausting and potentially dangerous, they would like to minimize the distance they have to swim as much as possible. In that regard, swimming directly from island A to B may not be the best option. Instead, it might be more beneficial to swim from A to C, then cross island C on foot, before swimming from C to B. The ideal travel plan could in fact involve a lot of islands.

You are given a collection of islands, modeled as simple polygons. Determine, for two given islands, the smallest possible total distance one needs to swim in order to get from the one island to the other. The total distance covered on land is of no importance.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with one integer n ($2 \leq n \leq 50$): the number of islands.
- one line with two space-separated integers s and d ($1 \leq s, d \leq n$; $s \neq d$): the islands that are the start and destination of the intended journey, respectively.
- per island:
  - one line with one integer m ($3 \leq m \leq 50$): the number of vertices of the polygon describing the island.
  - m lines with two space-separated integers x_i and y_i ($-10\,000 \leq x_i, y_i \leq 10\,000$): the coordinates of the i-th vertex of the polygon.

The polygons (islands) are non-self-intersecting and do not overlap or touch each other. The vertices of the polygons are given in counterclockwise order.

Output

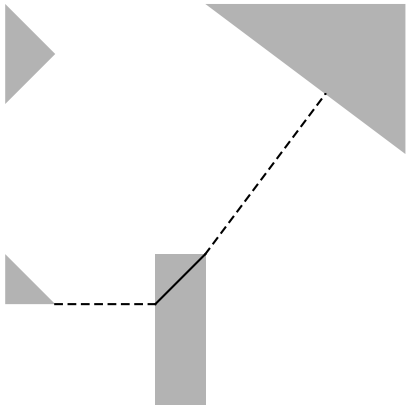
Per test case:

- one line with one floating point number: the minimum distance one needs to swim in order to get from the start to the destination, rounded to three decimal places.

The test cases are such that an absolute error of at most 10^{-6} in the final answer does not influence the result of the rounding.

Sample in- and output

Input	Output
2 3 1 3 4 0 0 1 0 1 1 0 1 4 2 2 3 2 3 3 2 3 4 4 0 5 0 5 1 4 1 4 1 2 3 0 0 10 0 0 10 3 40 60 80 30 80 60 3 0 40 10 50 0 60 4 30 -20 40 -20 40 10 30 10	2.828 60.000



The island group as described by the second test case in the sample input. Indicated is the optimal path from the bottom left island to the top right one, which is the path that minimizes the total distance covered over water (indicated by the dashed lines).

J Jumbled Letters

In the game of Scrabble you are given a number of tiles with letters on them. The challenge is to create a word using some (or all) of these tiles. Write a program which, with the help of a dictionary, determines the longest word you can make with the given letters. Those of you who know this game may note that we ignore the board upon which the letters are placed, as well as the value of the letters.

Input

The input file consists of:

- one line with an integer n ($1 \leq n \leq 100\,000$): the number of words in the dictionary.
- n distinct lines, each with one word consisting of between 2 and 10 lower-case letters.
- one line with an integer c ($1 \leq c \leq 10\,000$): the number of test cases.
- c lines, each with a string consisting of between 2 and 10 lower-case letters: the letters you have to make a word with.

The words in the dictionary are given in alphabetical order.

Output

Per test case:

- one line with the longest word from the dictionary you can make with the given letters. If there are multiple words, give the first one in alphabetical order. If there are no words you can make, output the word "`IMPOSSIBLE`".

Sample in- and output

Input	Output
6 algorithm balloon bapc code submit utrecht 5 abcdeop abcdeoq chuttep chutter iamhotgirl	bapc code IMPOSSIBLE utrecht algorithm

Almost blank page