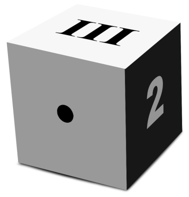
Problem A. The simplest problem about cubes

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	256 MiB



Yesterday Taja visited the museum. The tour was long and interesting, but the room she liked most contained collections of the cubes of the ten famous collectors. One of the cubes has attracted her attention a lot, but she forgot, who has possessed it. Nevertheless she memorized how three visible faces looked like, as well as criteria of each of the collector. You are to deduce the name of the collectors, who could possess such a cube, from this information.

Each cube has 6 faces. Each face has a single number written on it from 1 to 6, unique for each face. Numbers can be represented either by dots, or as a decimal or as roman numeral. Also each face has one of the following colors — Black, White, Green, Yellow, Skyblue, Red, Orange and Purple.

Here is the list of the names of the collectors and their corresponding criteria, which hold for the entire collection:

John	All numbers are represented as dots
David	Numbers are never written as roman numerals
Peter	All faces are white
Robert	Faces of the cube are either black or white
Mark	Odd numbers have white background, even numbers have black background
Paul	All prime numbers are written as decimal, and vice versa
Patrick	All face have the same color, but neither black nor white
Jack	All roman numerals are on the yellow background
Max	All faces have unique colors
Alex	Numbers of the same format have the same background, different formats have different colors

Input

Input contains three lines, describing visible faces of the cube.

First symbol of ith line c_i ($c_i \in \{B, W, G, Y, S, R, O, P\}$) — the color of ith face (Black, White, Green, Yellow, Skyblue, Red, Orange and Purple correspondingly). The, separated by space, follows the number written on the face, in one of the following formats:

1. From 1 to 6 «`.`» symbols (ASCII 46), which means that number is written with dots and the number equals to the number of this dots;
2. Decimal number from 1 to 6;
3. Roman numeral, written with «`I`» (ASCII 73) and «`V`» (ASCII 86) symbols.

It is guaranteed, that presented cube belongs to at least one collector.

Output

Output should contain a single line with name of collectors, which can own this given cube. Names should be written in any order and separated by space.

All names of the collectors should be from the following list: `John`, `David`, `Peter`, `Robert`, `Mark`, `Paul`, `Patrick`, `Jack`, `Max`, `Alex`.

Examples

input.txt	output.txt
`W ..`	`John David Peter Robert Jack Alex`
`W ...`	
`W ....`	
`B 2`	`David Robert Mark Jack`
`W 3`	
`B 6`	
`G 1`	`Patrick`
`G 2`	
`G V`	
`G 2`	`David Paul Jack Alex`
`G 3`	
`Y ....`	
`W .`	`Robert Mark`
`B 2`	
`W III`	

Problem B. Perfect gift

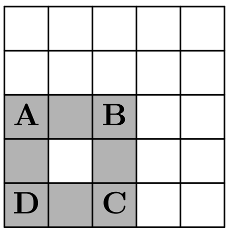
Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	256 MiB

Taja prepares a present for the birthday. As you might know, the best present is the one handcrafted by yourself. Recently she learnt cross-stitching and decided to make use of this skill.

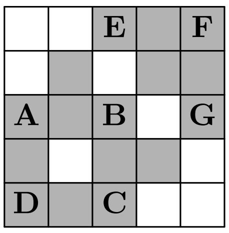
At home she only managed to find a canvas, which already had two crosses stitched on it. Don't panic — you can always complement it to the full picture. She had little experience, that's why she chose simple but nevertheless beautiful picture, which is parallelepiped. She wants to finish the present as soon as possible, thus number of new cross-stitches should be **the least possible**.

Parallelepiped on the **infinite** grid is drawn like this.

Let's draw a rectangle $ABCD$ with its upper left corner at A and lower right corner at C.



Then draw segments of equal length towards up-right from A, B and C — with ends at E, F, G correspondingly. Then add segments EF and FG.



All edges of the parallelepiped should be **at least** 3 **cells** long.

Input

First line of the input contains two integers x_1 and y_1 — coordinates of the first cross-stitch. Second line contains coordinates of second cross: x_2, y_2. Coordinates of the first cross-stitches are different. Axis OX is directed from left to right, and axis OY — from the bottom to the top. All numbers are within range $[0, 10^9]$.

Output

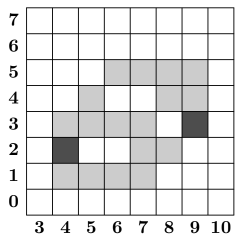
Output should contain single number — **the least** amount of required cross-stitches.

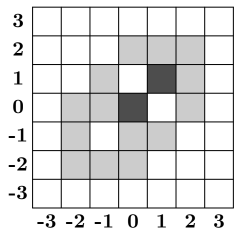
Examples

input.txt	output.txt
4 2 9 3	17
0 0 1 1	14

Explanation

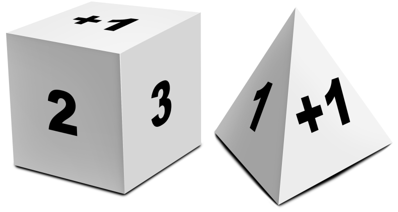
This pictures correspond to the samples:





Problem C. Casino

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	256 MiB



When Taja runs out of money, she goes to the casino. Recently a new game appeared at the casino, and Taja wants to master it. Help her.

Two parties of the game are croupier and visitor of the casino. Croupier has a single regular k-faced dice, which has all integers from 1 to k written on its faces. Croupier starts the game with rolling the dice once. Shown number determines amount of points gained by croupier.

To win, visitor has to gain more points, than croupier did. For this there's suggested a choice out of n options. Each option is a pair: the dice and number of its allowed rolls. Each face of each dice has some number written on it. This dice is rolled required number of times, all shown numbers are summed up and this sum is exactly the points gained by a visitor.

But some faces, in addition to numbers, have bonus marks. If shown face has bonus mark, then corresponding amount of points is added to the total, and visitor get additional dice roll. All faces of the same dice are pairwise distinct, which means there's no two identical bonus faces and no two identical ordinary faces. Each dice has at least one face without bonus mark. For every dice, the probability of each of its face being shown is the same.

In this problem it is required that for each possible amount of croupier's points from 1 to k you determine visitor's rolling option number, which leads to the maximal probability to gain stricly greater points than croupier did.

Input

First line of the input contains single integer n ($2 \le n \le 10$) — number of dice rolling options.

Next n lines contain descriptions of options in the following format.

First number c_i ($1 \le c_i \le 10$) — number of allowed rolls. Second number f_i ($2 \le f_i \le 12$) — number of dice faces. Next f_i numbers v_{ij} — numbers written on the faces. v_{ij} is either simply a number from 1 to f_i, meaning amount of points, or it can have additional plus sign «+» (ASCII 43) in front of the number, which is the bonus mark. For every dice, its numbers without plus sign are unique, all numbers with plus sign are unique, and there's at least one face without bonus mark.

Last line contains single integer k, which always equals $\max\limits_{1 \le i \le n} (c_i \times f_i)$.

Output

Output should contain k lines, each of which contains single integer b_i — number of the best option, which will allow to win with maximal probability by gaining more than i points (this probability shouldn't deviate from the right answer more than 10^{-9}).

Dice are numbered from 1 in the order thay are given in the input.

Examples

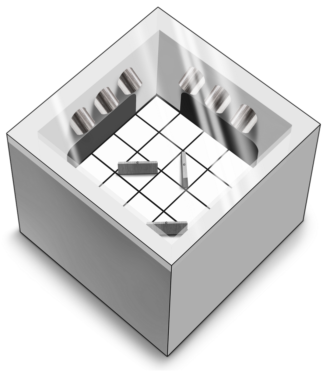
input.txt	output.txt
3	2
3 4 1 2 3 4	1
2 6 1 2 3 4 5 6	1
1 12 1 2 3 4 5 6 7 8 9 10 11 12	1
12	1
	1
	1
	3
	3
	3
	3
	2
2	2
1 4 1 2 +1 +2	2
1 6 1 +1 2 3 4 5	2
6	2
	1
	1

Explanation

Answer for the first sample could conain 1 on the first line, and the last could be any from 1 to 3.

Problem D. Puzzle

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	256 MiB



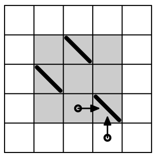
Once a puzzle has been gifted to Taja, and she still has no idea how to solve it.

The puzzle is a grid $n \times n$, with each row and each column containing **exactly one** separator, which is diagonal segment which starts in upper left corner and ends at lower right corner. Puzzle has a launch button, which launches the balls at integer time moments from the tubes, which are positioned at the **boundary of the grid**. Per one moment a ball moves to an adjacent cell. When a ball collides a separator it changes direction by 90°. A ball disappears if it crosses border line.

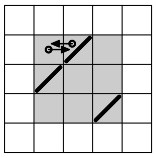
To solve a puzzle, one needs to rotate some separators 90° around their centers, in such a way that no two balls will ever collide inside the grid.

Two balls collide if:

1. They are at the same cell at the same moment (if the cell contains separator, then both balls should be at the same side).



2. They collided at the cells' boundary (boundary of the whole grid counts as well).



In this problem you are to find **any** solution of this puzzle.

Input

First line of input contains single integer n $(1 \le n \le 500)$ — grid size.

Second line contains n integers $(1 \le c_i \le n)$ — column number of ith separator, which has i as a row number. All column numbers are different.

Third line conatins single integer m $(1 \le m \le 10^4)$ — number of balls.

Each of the following m lines contains 3 integers x_i, y_i, t_i $(0 \le t_i \le 10^8)$, describing moments of balls' launches — at the moment t_i a ball will appear at (x_i, y_i) cell, which shares common side with the boundary of the grid. Moments are given in non-decreasing order of t_i. Coordinates (x_i, y_i) can be at one of the four following areas:

1. $x_i = 0$, $1 \le y_i \le n$;
2. $1 \le x_i \le n$, $y_i = 0$;

3. $x_i = n + 1$, $1 \leq y_i \leq n$;

4. $1 \leq x_i \leq n$, $y_i = n + 1$.

It is guaranteed that solution always exists.

Output

Output should contain single line of 0 and 1. ith symbol is 0, if ith separator doesn't require rotation, 1 — otherwise.

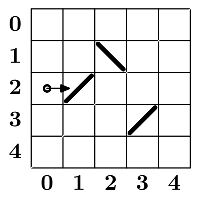
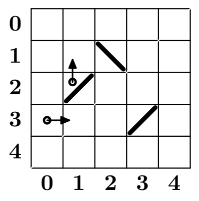
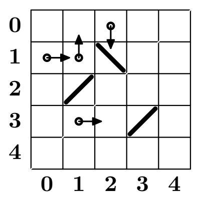
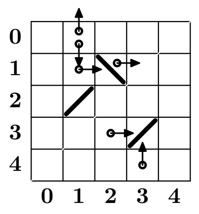
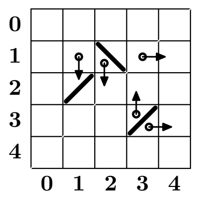
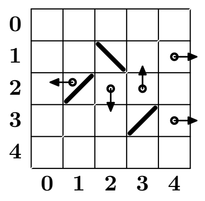
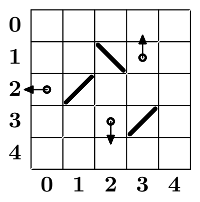
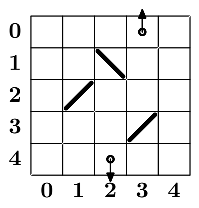
Examples

input.txt	output.txt

3
2 1 3
6
2 0 0
3 0 1
1 0 2
0 2 2
4 3 3
0 1 3
	011

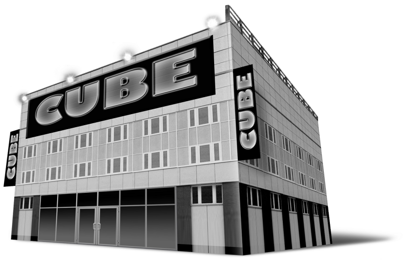
Explanation

Below are shown sample positions of the balls along the time.

0		**1**	
2		**3**	
4		**5**	
6		**7**	

Problem E. In the cube

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	3 seconds
Memory limit:	256 MiB



Taja likes to go to the cafe «In the cube» with her friends, since it has very convenient ordering system. To make an order, guest should walk to the automated stand and choose any dishes they likes. There are several such stands and they all are fixed at specific place inside the cafe.

In the cafe guests sit in front of tables, there are k tables. ith table cannot serve more than c_i persons. Uncomfortableness of the table position is the sum of the distances from this table to c_i automated stands closest to it.

Formally, cafe is the grid $(0, 0) - (5000, 5000)$. Each cell (x, y) $(0 \le x, y \le 5\,000)$ can contain either single automated stand or single table or nothing.

The distance between cells (x_1, y_1) and (x_2, y_2) equals to $|x_2 - x_1| + |y_2 - y_1|$.

You are to arrange the tables in such a way, that total sum of uncomfortablenesses for all tables should be minimal.

Input

First line of the input contains two integers n and k ($1 \le n \le 18$, $1 \le k \le 200$) — amount of automated stands and tables correspondingly.

Following n lines contain coordinates of ith stand: two integers x_i and y_i ($0 \le x_i, y_i \le 5\,000$).

Next of each k lines contain single integer c_j ($1 \le c_j \le n$) — number of seats at jth table.

Output

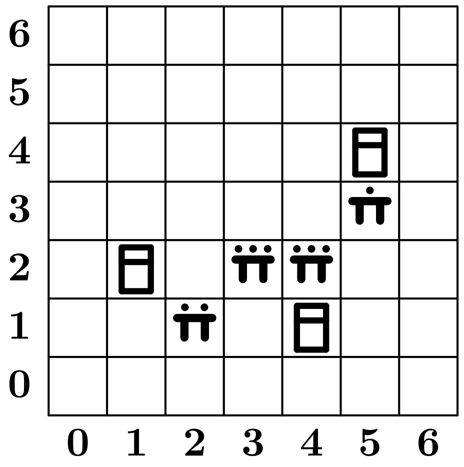
Output should contain single integer — minimal total uncomfortableness.

Examples

input.txt	output.txt
3 4	20
1 2	
4 1	
5 4	
1	
2	
3	
3	
2 10	16
0 0	
5000 5000	
1	
1	
1	
1	
1	
1	
1	
1	
1	
1	

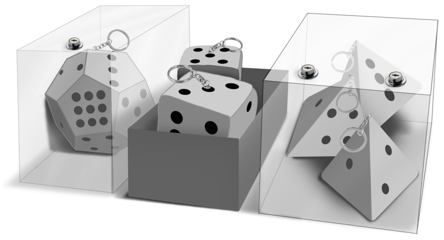
Explanation

Possible arrangement of the tables for the first sample looks like this:



Problem F. Toy store

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	256 MiB



Taja often went by the toy store and looked at electoronic table near the store window which displayed two integers. Store window shows several kinds of toys, but numbers on the table may be out of sync with actual number of different kinds. It turned out that, not every toy from the store window can be bought, because they do not remove toys as soon as possible, but only after some time past the purchase. For different kinds of toys this time can be different.

There are n kinds of toys. For each kind of toy it is known that initial amount of toys is c_i and time is t_i minutes after the purchase, after which the toy is removed from the window. Each minute the following happens:

- removal of toys from the store window, that were purchased corresponding amount of minutes ago;
- the electronic table is updated;
- new customer comes and **necessarily buys some toy, that is remaining in stock**.

Taja has been always interested the meaning of the numbers on the electronic table and recently she found it out. Both numbers show how many kinds of toys can be bought in the store, but first one shows number of kinds, **possibly** in stock up to the current moment, and second one is number of kinds, that are in stock up to the current moment **for sure**. Also Taja is interested how much is this table informative for customers. That's why she needs a program, that will model behaviour of the customers and update the table.

Your task is: for each minute calculate electronic table numbers.

Input

First line of the input contains single integer n ($1 \le n \le 10^5$) — number of kinds of the toys.

Each of the following n lines contains two integers c_i and t_i ($1 \le c_i \le 10^5$, $1 \le t_i \le 100$) — number of toys of ith kind and time, after which the toy will be removed out of the store window after the purchase correspondingly.

Next string contains single integer k ($1 \le k \le 10^5$) — number of customers.

Each fo the following k lines contains integer q_i and q_i integers $p_1, p_2, ..., p_{q_i}$ — number of toys, that were removed at ith minute and numbers of these toys.

Output

Output should contain k lines, each of which contains two integers a_i and b_i — numbers on the electronic table at moment of the beginning of ith minute correspondingly.

Examples

input.txt	output.txt

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

Explanation

In the above example, the store window contains one toy of the first kind, two toys of the second kind and three toys of the third kind, which were removed after 2, 1 и 3 minutes correspondingly after the purchase. Numbers on the table shoud change in the following order:

- 3/3: there were no customers before the first one, he can buy any toy.
- 3/2: first customer could possibly buy the toy of first kind, thus there's no confidence, that second customer can buy it.
- 3/2: since the toy of neither the first kind nor second kind has been removed from the store window, then it means that first customer purchased the toy of the third kind. What was purchased by the second customer is impossible to decide yet.
- 2/2: no toys of the first kind anymore.
- 2/2: no toy has been removed from the store window, which means that previous customer bought a toy of the third kind.
- 1/1: There's only one toy of the third kind remaining.

Problem G. Gifts delivery

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	256 MiB

A trouble has happened at Taja's work: a truck driver got sick, while there's is an urgency to deliver gifts from one store to another. Fortunately, currently she has a break, and this store is situated on the same street so her skill to drive only forward with constant speed v_1 is quite sufficient, to help the situation.

But one of the crossroads on the way to the store has broken traffic lights, and now there is traffic guard, who is not supposed to leave his place.

At some moment he noticed the truck moving towards him and having no intention to steer aside. And he's not supposed to move — he will be penaltized for that — nevertheless he will have to. That's why traffic guard wants to allow the truck bypass in such a way that **he will minimize his time of being off his initial position**. Traffic guard can move in any way, but his speed cannot exceed v_2.

Regard the truck as a rectangle and traffic guard as a dot. It is required that dot should never be strictly inside the rectangle and time, during which the dot isn't at (p, q) (its initial position), should be minimal possible.

Input

First line contains 6 integers a, b, p, q, v_1, v_2 ($1 \le a \le 100$, $0 \le b \le 99$, $-a < p < a$, $b < q \le 100$, $1 \le v_1, v_2 \le 100$). Initially upper left corner of the truck is at $(-a, b)$, lower right corner is at $(a, 0)$. Traffic guard initially stands at the point (p, q). Truck moves towards increasing of the second coordinate with constant speed v_1. Maximal speed of the traffic guard is v_2. If $b = 0$, regard the length of the truck being as small as required.

All distances are measured in meters, speed is measured in meters per second.

It is guaranteed that all the values are such that answer won't exceed 10 000.

Output

Output should contain single real number — the least possible time, when the traffic guard will be absent at (p, q) point. Answer should be given with absolute or relative error that doesn't exceed 10^{-6}.

Examples

input.txt	output.txt
`4 0 1 5 1 1`	`6`
`3 2 -1 10 5 2`	`2.306019375`

Explanation

In the first sample it would be optimal to wait for 2 seconds, then move for 3 seconds to the right with maximal speed, and then move backwards-left with maximal speed.

Problem H. Game with dices

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MiB
Queries limit:	7 500

This problem is interactive.

At her loft, Taja found an ancient tabletop game, which she managed to win only from time to time. Show Taja, how to win this game with a guarantee.

Game equipment consists of round piece with a radius of 1, which has an arrow drawn on top of it, two dices and 360 stickers. Every sticker has unique integer written on it, from 0° to 359°.

Before starting to play one should mark a point on a table, then place round piece on the table, then choose 12 different stickers, and 6 of them stick on the first dice, and another six on the second dice. The goal is to cover the marked point with the piece. It should be achieved by making turns with the following rule. First, player rolls one of the dice and rotates the piece counterclockwise by amount of degrees written on top of the dice. Then the piece moves towards the arrow by the distance of 10.

Co-ordinates of the marked point is always $(0, 0)$. Starting position of the center of the piece is (x, y) and satisfies the following constraint:

$$2 \le \max(|x|, |y|) \le 500$$

Number of queries for this problem equals to the number of made turns.

Interaction protocol

Interactor starts with giving coordinates of the center of the piece and direction of its arrow. Then your program should respond with numbers, sticked on both dice. Then for each number of dice given by your program, interactor outputs amount of degrees shown by the dice and says whether the piece reached the goal. If the piece has covered the marked point, then your program should terminate. Otherwise, interactor outputs resulting position of the piece and its arrow, thus initiating next turn.

Output

First two lines of the output should contain 6 integers each, ranging from 0 to 359 — stickers for first and second dice correspondingly. All integers in these lines should be unique.

Following lines should only contain either 1 or 2 — number of the dice to roll.

Don't forget to flush the standard output after printing each line.

Input

Input consists of quadruples of lines:

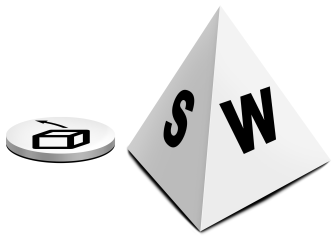
1. x, y — coordinates of the center of the piece;
2. v_x, v_y ($v_x^2 + v_y^2 = 10$) — arrow direction of the piece;
3. d — amount of degrees shown on the dice (each side of the dice is shown with the same probability);
4. «`Yes`» — the piece has covered $(0, 0)$ point, «`No`» — otherwise.

Examples

standard output	standard input
	10.000000000 -10.000000000
	0.000000000 -10.000000000
180 96 250 187 319 6	
295 152 82 90 32 334	
1	
	180
	No
	10.000000000 0.000000000
	0.000000000 10.000000000
2	
	90
	Yes

Problem I. Game with coins

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds (3 for java)
Memory limit:	256 MiB
Queries limit:	60 000



This problem is interactive.

Taja can easily win this game, but all of her friends couldn't. Now she offers you to play this game.

Game equipment consists of field $n \times n$ ($5 \leq n \leq 40$), the piece, two coins (`COIN1`, `COIN2`), two dices with cardinal directions (`DICE1`, `DICE2`), and a lot of one cell size blocks.

Name	Number of faces	Faces
`COIN1`. Move coin	2	SLIDE, RAM
`COIN2`. Modification coin	2	PLACE, REMOVE
`DICE1`. First dice with directions	4	`N` (North), `S` (South), `W` (West), `E` (East)
`DICE2`. Second dice with directions	8	`N` (North), `S` (South), `W` (West), `E` (East), `NW` (Northwest), `NE` (Northeast), `SW` (Southwest), `SE` (Southeast)

Before the beginning of the game the piece and some blocks are placed on the field. Then the player makes moves such kind. At first, the player chooses a coin and tosses it for determine an action. Then he chooses one of the dices and rolls it for determine direction *dir*. After that one of the four action happens:

Coin string	Action
`SLIDE`	Move the piece along empty cells in the direction of *dir*, until the piece collides a block or maze border
`RAM`	Move the piece along empty cells in the direction of *dir*, until it collides a first block. Then move the piece and the block in the same direction, until moving block collides another block or field border
`PLACE`	If adjacent cell in the direction *dir* next to the piece is empty and is in the field, place a block on this cell
`REMOVE`	If adjacent cell in the direction *dir* next to the piece contains a block and is in the field, remove the block from this cell

The goal is to put the piece on the finish cell.

One query for this problem is one coin toss and one dice roll.

Interaction protocol

At first, interactor gives the size of the maze, the maze and coordinate of the finish cell. Then the following 4-step actions are happens:

1. Jury's program displays coordinates of the piece or inform that the piece has reaches the finish cell.

2. Your program displays the coin name.
3. Jury's program displays the name of the action on the coin.
4. Your program displays the dice name.
5. Jury's program displays the name of the direction on the dice and additional information for action `"RAM"`.

Output

The standard output should consist of pair of lines: coin name and dice name. Coin name is the string `"COIN1"` or `"COIN2"`. Dice name is the string `"DICE1"` or `"DICE2"`.

Don't forget to flush the standard output after printing each line.

Input

First line of the standard input contains one integer number n — the size of the maze. Next n lines contains n characters `"."` (ASCII 46) or «`#`» (ASCII 35), denoting empty cell and cell with block respectively.

Next line contains two integer numbers r_f and c_f — row number and column number of the finish cell. The upper-left corner is $(1, 1)$, the bottom-right is (n, n). Finish cell and initial cell are empty.

Next groups describes each move:

- The first line of an group contains two integer numbers r, c — row number and column number of the piece, or $(-1, -1)$, if the peace reaches the finish cell.
- Next line contains action name shown on the coin.
- Next line contains direction name shown on the dice.
- If the current action is `"RAM"`, next two strings contains two integer numbers r_1, c_1 and r_2, c_2, denotes then the piece collides a block with coordinates (r_1, c_1) and moves that block to the cell (r_2, c_2).

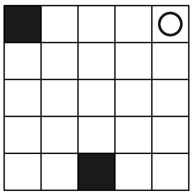
The north direction corresponds to decreasing row number. The south to increasing row number. The west direction to decreasing column number. The east direction to increasing column number.

Each side of any coin or dice is shown with the same probability.

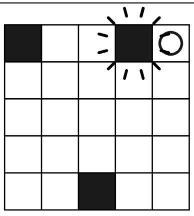
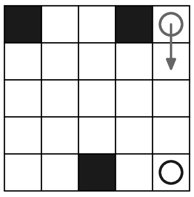
Examples

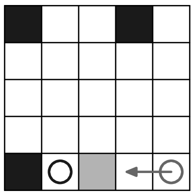
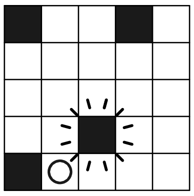
standard output	standard input
	5
	#....
	.....
	.....
	.....
	..#..
	4 3
	1 5
COIN2	PLACE
DICE1	W
	1 5
COIN1	RAM
DICE1	S
	6 5
	6 5
	5 5
COIN1	RAM
DICE1	W
	5 3
	5 1
	5 2
COIN2	PLACE
DICE2	NE
	5 2
COIN1	RAM
DICE2	NE
	4 3
	2 5
	3 4
COIN2	REMOVE
DICE2	NE
	3 4
COIN2	PLACE
DICE1	S
	3 4
COIN1	SLIDE
DICE1	W
	3 1
COIN1	RAM
DICE1	S
	5 1
	5 1
	4 1
COIN1	SLIDE
DICE1	E
	-1 -1

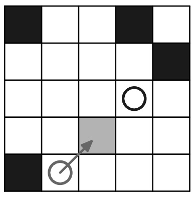
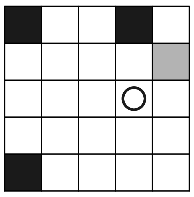
The initial state of the maze for the sample is

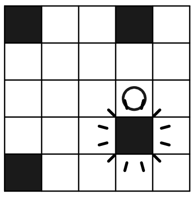
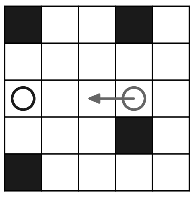


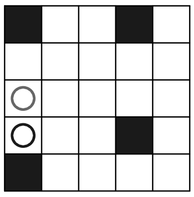
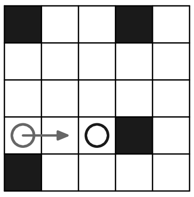
And the representation of the path is shown below:

`COIN2 → PLACE`
`DICE1 → W`
		`COIN1 → RAM`
		`DICE1 → S`
			

`COIN1 → RAM`
`DICE1 → W`
		`COIN2 → PLACE`
		`DICE2 → NE`
			

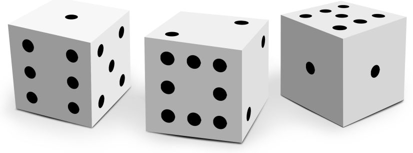
`COIN1 → RAM`
`DICE2 → NE`
		`COIN2 → REMOVE`
		`DICE2 → NE`
			

`COIN2 → PLACE`
`DICE1 → S`
		`COIN1 → SLIDE`
		`DICE1 → W`
			

`COIN1 → RAM`
`DICE1 → S`
		`COIN1 → SLIDE`
		`DICE1 → E`
			

Problem J. The hardest dice problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 MiB



This problem is interactive.

Taja plays her own game very well. You have a unique opportunity to play along and try to win.

The game equipment consists of two identical sets of n ($2 \le n \le 10$) 6-faced dice, each face of which has a number from 1 to 100 written on it. Players are playing simultaneously and independently with zero knowledge about each other's states of the game.

You play the game in the following way. You choose any dice from the set and roll it. You can either accept the number shown (this will be amount of points you receive) or roll another dice, but you will additionally receive 1 penalty point in this case. You never roll the same dice twice during a single play. Your total score equals to the difference between last number and number of repeated rolls. The game ends, when both players decide to accept the shown number.

Since Taja played this game for several years, she will play slightly weaker. You will be considered a winner in the game if your score **greater or equal** to her score. Also Taja will stick to same strategy during single testcase: she will roll dices always in the same order. And she will decide whether to repeat the roll in the following way: if, continuing to roll dices in predetermined order, she can score higher than scores for the last dice roll with probability of at least 50% (taking penalty into account), then she continues to play, and stops otherwise.

In this problem you are to play with Taja 10 000 games and win at least 5 000 of them.

Interaction protocol

First, interactor outputs description of dice. Then your program should play with interactor 10 000 games. Each game goes as follows. Your program outputs number of dice to be rolled. Interactor responds with your score, including penalty. Then your program answers, whether it accepts the shown number. After the end of the game, interactor outputs result of the game — whether you win or not. Then the next game begins.

Output

To roll the dice output the line containing single integer from 1 to n — number of dices. After each roll you should output line containing either string «`Yes`», if you accept your current score, or «`No`» otherwise.

Don't forget to flush the standard output after printing each line.

Input

First line of the input contains single integer n — number of dice.

Each of next n lines contains 6 integers from 1 to 100 — numbers written on the faces of ith dice.

For each dice roll input contains single integer — the shown number. All faces are shown equiprobably.

When round ends, input contains single line — «`Win`», if you program won, or «`Lose`» otherwise.

Examples

standard output	standard input
	3
	1 2 3 4 5 6
	2 2 2 8 8 8
	1 1 1 7 7 7
1	
	1
No	
2	
	1
No	
3	
	5
Yes	
	Lose
2	
	8
Yes	
	Win

Explanation

Example shows only two plays. Real testing will go through all 10 000 games.

In this testcase Taja rolls dices in the same order as they are given in the input.