Problem 1. GUI

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 megabytes

Your company is developing a new window-based GUI. The existing implementations of this interface type are fatally flawed: someone else made them. Windows are rectangles made of pixels. Sides of windows are parallel to screen sides. You are to write a program determing the relative positions of windows. According to your task, the program receives positions of two windows A and B as input and produces a verdict on their relative position. The following statements are possible:

- `A in B` — all pixels of the window A are pixels of the window B;
- `B in A` — all pixels of the window B are pixels of the window A;
- `Separate` — A and B do not share any pixels;
- `Intersect` — for all other cases;

It is guaranteed that the positions of the windows A and B are not the same.

The window position is defined by the coordinates of two pixels: the upper left corner and the lower right corner. The pixel with the coordinates (0, 0) is located in the lower left corner of the screen. The coordinate values ascend from left to right and downwards.

Input

The first line of the input file contains a single integer N — the number of sets of input data for your program ($1 \le N \le 1000$). It is followed by N pairs of lines.

Each set of input data is described with two lines: the first line gives the position of the window A, and the second line gives the position of the window B. For each window there are four integers: the coordinates X_l, Y_t of the upper left pixel and the coordinates X_r, Y_b of the lower right pixel ($0 \le X_l \le X_r \le 10^5$, $0 \le Y_t \le Y_b \le 10^5$).

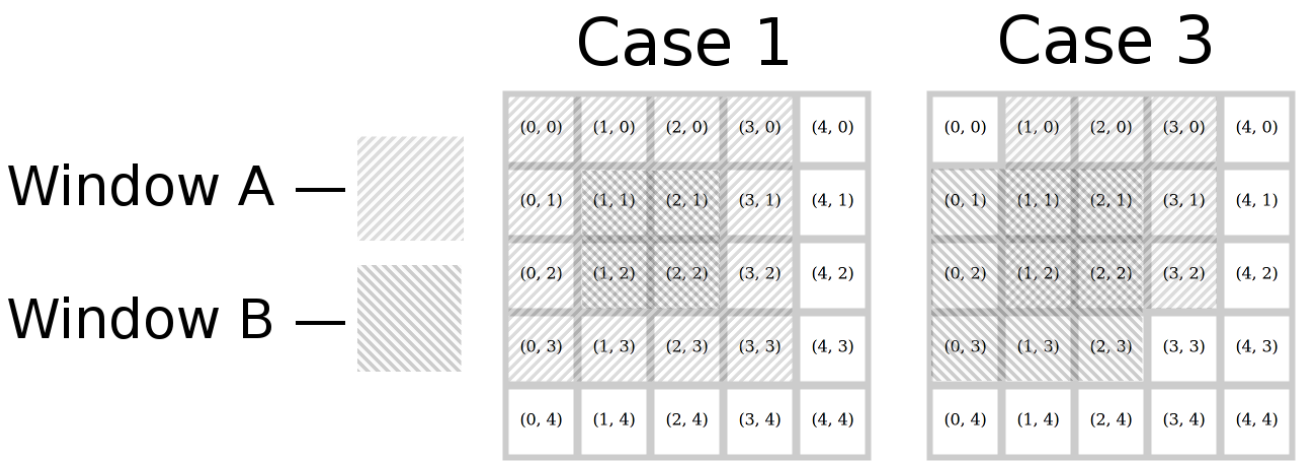
Output

For each set of input data print a verdict on the relative position for the given windows.

Example

input.txt	output.txt
3 0 0 3 3 1 1 2 2 0 0 9 9 10 0 19 9 1 0 3 2 0 1 2 3	B in A Separate Intersect

Illustration



Problem 2. Searching on the Cube

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	256 megabytes

Two friends are playing a new board game they just invented. The lead thinks of a cube size and the initial position of the robot on the cube. The player then tries to find the treasure, using the "hotter / colder"principle, guiding the robot blindly.

The game field is a cube of size $N \times N \times N$; each of its sides is divided into $N \times N$ cells. According to the game rules, the lead can set the cube size N in the range between 3 and 300. The lead places a treasure in one of the $6N^2$ cells on the cube surface; the treasure stays in the same cell throughout the game. The lead also places the robot into one of the cells; the robot faces one of the four possible directions parallel to one of the cell sides.

The game is played step by step. In each step, the player gives a single command to the robot. There are four types of commands:

- `left` — the robot turns 90 degrees left, remaining in the same cell.
- `right` — the robot turns 90 degrees right, remaining in the same cell.
- `forward` — the robot moves one cell forward.
- `dig` — the robot digs in the cell where it is positioned.

The left/right direction is defined as seen from the outside of the cube.

When the `forward` command is executed, firstly, the side of the robot's cell it looks at is determined. Then the robot moves towards this side to the adjacent cell. If after such move robot remains on the same side of the cube, its direction doesn't change. If the robot moves to the adjacent face of the cube, its direction naturally rotates by 90 degrees around the cube edge it walks through. A simple rule holds true: if after the execution of the `forward` command the robot is turned 180 degrees, and the `forward` command is executed again, the robot returns to the previous cell.

The player does not see the playing field: he or she tells the commands to the lead blindly, and the lead executes these commands. Each time a `forward` command is executed, the lead tells the player one of the three words:

- `closer` — the robot is now closer to the treasure.
- `farther` — the robot is now further from the treasure.
- `same` — the distance between the robot and the treasure has not changed.

The distance from the robot to the treasure is the euclidean distance between their cell centers, i.e. the length of the segment between these centers. Note that the distance is calculated directly through the cube and not along its surface. If this value decreases, the lead says `closer`, and if it increases — `farther`.

Write a program which will play this game as the player. You cannot make mistakes: when the `dig` command is executed, the robot must be in the treasure cell.

Interaction protocol

This is an interactive problem. Instead of file input-output, you will work with a special program — an interactor. You will interact with this program via the standard input-output streams.

Your program sends robot control commands to the output stream. Each command is defined by the words `left`, `right`, `forward` or `dig` (see description of commands above). After each `forward` command printed by your program one of the following words is sent to the input stream: `closer`, `farther` or `same` (see description of results above).

Once you print `dig`, the game is over. If at this moment the robot and the treasure are in different cells, your solution will get Wrong Answer. Also, the number of commands must not be greater than $100N$, otherwise the solution will get Wrong Answer.

Make sure you print the newline character and clear the buffer of the output stream (the `flush` command) after each printed command. Otherwise the solution may be given a Timeout verdict.

Example

stdin	stdout
farther closer closer farther closer	forward left left forward forward forward right right forward dig

Problem 3. Mirrors

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	7 seconds
Memory limit:	512 megabytes

The dream has come true! No more fleeing from the widow's rage through the corridors. From now on, Madame Gritsatsueva is the main character of the "Reflect me!" game.

According to the game rules, she is moving in a rectangular game field of square cells of size $N \times M$, constantly reflecting either from the field borders or from rotating or stationary double-sided mirrors. Each mirror is a unitary segment with its center located in the center of a game field cell. Madame herself is much smaller than a mirror or a unitary cell and thus she can be considered a point, which, at any integer moment of time, is located in the center of a cell.

Every second Madame moves to an adjacent cell, unless she reflects from the field border. If she reaches a cell with a mirror, she changes her direction according to the reflection rule, i.e. "the angle of incidence equals the angle of reflection", that is: 90 degrees if the mirror is at 45 degrees to her direction, 180 degrees if the mirror blocks her way, i.e. is perpendicular to her direction. If the mirror does not block her movement, i.e. is parallel to her movement, it doesn't affect her further moves. If the widow finds herself in a boundary cell, her direction changes to the opposite direction. In this case, she stays in the boundary cell for two adjacent moments of time (think of it this way: she moves from the center of the cell to the margin for a half of a second, reflects instantaneously and moves back from the margin for another half of a second).

Madame may start at any nonnegative moment of time and in any direction from one of the acceptable start points. Her goal is to end up in the exit point with the coordinates (r_0, c_0) as soon as she can.

Input

The first line of the input file contains six integers: N, M, K, S, r_0, c_0, where N, M — vertical and horizontal field dimensions ($1 \le N, M \le 10^9$), K — the number of possible starting points ($1 \le K \le 10^3$), S — the number of mirrors ($0 \le S \le 10^3$), r_0, c_0 — the number of row and column where the exit point is located ($1 \le r_0 \le N$, $1 \le c_0 \le M$).

The next K lines contain the coordinates of the acceptable start points, with two integers in each line r_j and c_j ($1 \le r_j \le N$, $1 \le c_j \le M$).

The remaining S lines contain descriptions of the mirrors: five integers r_i, c_i, a_i, t_i, p_i in each. Here r_i, c_i are the numbers of row and column of the cell where the i-th mirror is located ($1 \le r_i \le N$, $1 \le c_i \le M$).

The number a_i describes the position of the mirror at the start moment ($0 \le a_i \le 3$). If $a_i = 0$, then the mirror is positioned vertically, and with each increment of the value a_i by 1 it turns 45 degrees clockwise. Thus if $a_i = 1$, then the mirror segment is directed from the lower left corner to the upper right corner; if $a_i = 2$, the segment is horizontal, etc.

The values t_i and p_i define the time of the first rotation of the mirror 45 degrees clockwise and the period of each consecutive turn, respectively ($1 \le p_i \le 7$, $1 \le t_i \le p_i$). Thus the mirror turns in the following moments of time: t_i — the first turn, $t_i + p_i$ — the second turn, $t_i + 2p_i$ — the third turn etc. Note that at the moments t_i, $(t_i + p_i)$ the mirror has already moved to a new position, and at the moments $(t_i - 1)$, $(t_i + p_i - 1)$ it is still in the previous position. If the mirror

is stationary, the last two numbers t_i and p_i in the description equal -1. It is guaranteed that all positions of starting points, mirrors and exit points are pairwise different, which includes that no mirror is located in the start point. The field orientation and row and column numbering are such that row numbers increase while moving down the field, and column numbers increase while moving right.

Output

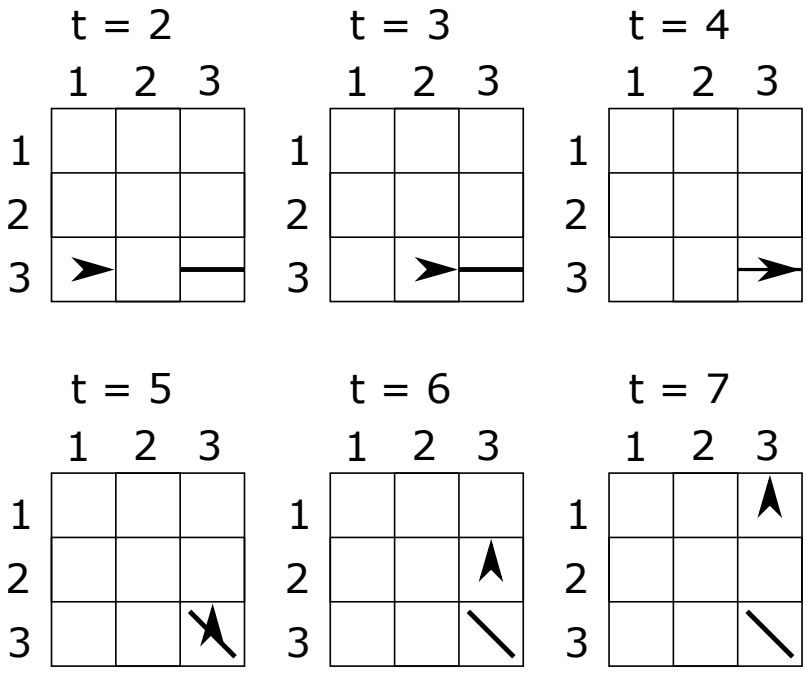
The output file must contain a single integer: the earliest time Madame can reach the exit point r_0, c_0, or -1, if that is impossible.

Example

input.txt	output.txt
3 3 1 1 1 3 3 1 3 3 2 5 7	7
3 3 1 2 3 2 1 1 1 3 2 5 7 3 3 2 -1 -1	-1

Example explanation

The picture below represents the first sample. An arrow shows a position and a direction of Madame, the segment in a cell $(3, 3)$ stands for a mirror. Madame starts at the time $t = 2$ to the right. At the time $t = 4$ she passes by the mirror and at the time $t = 4.5$ reflects from the margin, after that she reflects from the rotated mirror at the time $t = 5$. At the time $t = 7$ Madame reaches the exit point.



Problem 4. Roads to cinematography

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 megabytes

After the fiasco in Santa Carolina Mr. John First and his friends opened a new shrine of cinematograpy. Their new promotion location was chosen not far from a major railroad. Black Jack offered to build roads to the nearest stations where passenger trains stop.

Mr. John First drew a coordinate system. The origin point was their current location, and the axes X and Y were oriented to the east and north, respectively. Stations were marked as points with the coordinates (x_i, y_i), where $i = 1, 2, \ldots, n$. The railway passes in the southeast direction, so the following inequations hold true:

$$\begin{cases} 0 \le x_1 \le x_2 \le x_3 \le \ldots \le x_{n-1} \le x_n \\ 0 \le y_n \le y_{n-1} \le y_{n-2} \le \ldots \le y_2 \le y_1 \end{cases}$$

Diana insists that each road should be oriented strictly at one of the cardinal directions, i.e. that the corresponding segment should be oriented to one of the coordinate axes. Jack says that roads must be built so that each station is connected by cinema directly or indirectly with those roads (without using the railroad) and that the distance between each station and the cinema is minimal. The distance must be minimal provided that the roads are parallel to the coordinate axes.

Roads cost money, and Mr. John First wants the total length of the roads to be minimal, provided that Diana's and Jack's conditions are met. The money they'd save could be spent on new films.

Input

The first line of the input file contains an integer n — the number of stations ($1 \le n \le 500$). Each of the following n lines contains two integers x_i, y_i — the coordinates of a station ($0 \le x_i, y_i \le 10^6$).

It is guaranteed that the sequence (x_i) is nonstrictly ascending and the sequence (y_i) is nonstrictly descending. No two stations are located in the same point. No station is located in the origin point.

Output

In the first line of the output file, print two integers: k — the number of roads to be built ($1 \le k \le 2000$) and A — the total length of these roads. Each of the following k lines must describe a road.

A road description must contain four integers x_1, y_1, x_2, y_2 — the coordinates of the end points of the road ($x_1 \ge x_2$, $y_1 \ge y_2$). One of the following two statements must hold true: either $x_1 = x_2$, or $y_1 = y_2$.

It is guaranteed that an optimal layout exists which meets the limitations of the output data format. If several solutions are possible, print any of them.

Example

input.txt	output.txt
3 0 9 2 4 5 1	4 16 0 9 0 4 2 4 0 4 5 1 0 1 0 4 0 0

Problem 5. Geometric solver

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 megabytes

Kedas is taking part in the development of a new geometric solver. His assigned task is to implement the removal of excessive constraints on a two-dimensional sketch. Help Kedas solve the task.

For this problem, assume there are only N line segments $L_1, L_2, \ldots, L_n$ in the sketch and a set of constraints imposed. Each *constraint* is a statement about some objects of the sketch. There are four types of constraints:

- `hor` i — the segment L_i is horizontal (the Y-coordinates of its endpoints are the equal).
- `vert` i — the segment L_i is vertical (the X-coordinates of its endpoints are the equal).
- `par` i j — the segments L_i and L_j are parallel.
- `perp` i j — the segments L_i and L_j are perpendicular.

A *solution* of the sketch is a placing of all $2N$ endpoints of all sketch segments on the coordinate plane, such that:

1. all sketch constraints are satisfied, i.e. the corresponding statements are true.
2. each segment has a non-zero length, i.e. its endpoints do **not** coincide.

The direction of a segment is considered irrelevant.

Excessive constraint can negatively affect the work of numerical algorithms of the geometric solver, so they need to be removed to the possible extent. A subset of sketch constraints is called *excessive* if its removal does not affect the set of solutions of the sketch. If the set of solutions of the sketch is empty, the sketch is called *inconsistent*, otherwise it is called *consistent*.

In this problem, find whether the sketch is consistent, and if it is, find the excessive set of constraints of the maximum size.

Input

The first line of the input file contains two integers: n — the number of line segments in the sketch ($1 \le n \le 10^5$), m — the number of constraints in the sketch ($0 \le m \le 10^5$). Each of the following m lines describes a single constraint.

Each constraint is described in the way shown above in the problem statement: first comes the word (`hor`, `vert`, `par` and `perp`) defining the type, followed by the space-separated number i of the first argument. If the type equals `par` or `perp`, then it is followed by another space-separated number j of the second argument. In this case $1 \le i \ne j \le n$.

Segments are numbered from 1 to n, constraints are numbered from 1 to m in the order they are described. The constraints may have completely identical descriptions.

Output

In the first line of the output file, print the word `consistent`, if the sketch is consistent, or the word `inconsistent`, if the sketch is inconsistent. If the sketch is inconsistent, do **not** print anything else.

If the sketch is consistent, the second line must contain a single integer k — the number of

constraints in the discovered excessive set ($0 \le k \le m$). The third line must contain k different integers — the numbers of constraints in this set in any order.

If there are several possible answers, print any of them.

Examples

input.txt	output.txt
2 4 vert 1 hor 2 vert 1 perp 2 1	consistent 2 4 1
2 4 vert 1 hor 2 vert 1 par 2 1	inconsistent

Example explanation

In the first example, there are two vertical constraints imposed on the segment L_1; apparently, one of them can be discarded without affecting the solution set. The perpendicularity of the segments L_1 and L_2 is also excessive, because one of the segments is vertical and the other is horizontal. If three constraints are removed, it is always possible to provide a solution where one of the segments is **not** parallel to the coordinate axes, i.e. the set of solutions of the sketch increases.

In the second example, the sketch is inconsistent, because the constraint 4 states that the segments L_1 and L_2 are parallel, while the constraints 1 and 2 stipulate that they are perpendicular.

Problem 10. Tickets

Input file:	input.txt
Output file:	output.txt
Time limit:	1 second
Memory limit:	256 megabytes

Kisa and Osya are sitting on their chairs and playing a curious game. Each of them is holding a tram ticket with a six-digit number from 000000 to 999999. They take turns to tell the digits on their tickets, from left to right, and after each pair the loser — the one whose digit is smaller — grants another one's wish.

Print the number of wishes each of them grants.

Input

The first line contains the number of Osya's ticket, and the second line contains the number of Kisa's ticket. Each number consists of six digits in the range from 0 to 9. The number may have leading zeroes.

Output

In the first line output the number of wishes Osya will grant; in the second line output the number of wishes Kisa will grant.

Example

input.txt	output.txt
013936 132946	3 1

Example explanation

Osya loses with the first, second and fifth digit; Kisa loses with the third digit.

Problem 11. Logarithm smoothing

Input file:	input.txt
Output file:	output.txt
Time limit:	1 second
Memory limit:	256 megabytes

The life path of Vasisualiy Lokhankin is all curves. It looks like a logarithm (well, multiplied by a coefficient). In his free time, brooding about the tragic fate of the Russian intelligentsia and the great truth, he decided to take a closer look at his trajectory. In fact, he decided to substitute it with something more adapted for the masses - that is, a polyline. Naturally, he wants the polyline to be as similar to the initial logarithm as possible. On the other hand, the number of segments in the polyline must not be greater than n.

Let's take a more formal look at this problem. A function defined on the interval $[a, b]$ is a polyline with no more than n segments if its graph can be split into n parts, each of the parts being a line segment.

Let $B_n[a, b]$ denote the class of continuous polylines defined in the interval $[a, b]$ and having no more than n segments. The distance between two functions $f(x)$ and $g(x)$ on the segment $[a, b]$ is the value:

$$D(f, g) = \max_{x \in [a,b]} |f(x) - g(x)|$$

Find a polyline from the class $B_n[a, b]$ with the minimum distance to the function $f(x) = c \ln x$.

Input

The first line of the input file contains a single integer m — the number of test tasks($1 \le m \le 20$).

It is followed by m lines each containing four numbers n, c, a and b, where n is the number of segments in the polyline($1 \le n \le 20$), c is the factor multiplying the logarithm ($1 \le c \le 10^4$), a, b are the left and right endpoints of the segment in question, respectively ($10^{-2} \le a < b \le 10^2$). n and c are integers, and a, b are real numbers given with no more than two digits after the decimal point.

Output

The output file must contain m lines each containing a single real number — the minimum possible deviation of the polyline with no more than n segments from the logarithm, the answer to the corresponding test task. The absolute or relative error of your answer must not be greater than 10^{-8}.

Example

input.txt	output.txt
2 1 1 1 10 1 1 0.5 7	0.309517460 0.398765993

Problem 12. Outer space signals

Input file:	input.txt
Output file:	output.txt
Time limit:	1 second
Memory limit:	256 megabytes

The outer space research group is still busy searching for extraterrestrial intelligence. A while ago, two signals came from the outer space, and presumably they weren't just noise. The researchers named the signals X and Y. They encoded both signals as lines consisting of lowercase Latin letters.

Recently, the researchers got lucky again — they received yet another signal. They named the signal Z and also encoded it as a string of lowercase Latin letters. They are facing a task: to find out if Z is essentially different or it is just a combination of the already known signals X and Y.

The signal Z is considered to be a combination of the signals X and Y if and only if it contains nonintersecting occurrences of the signals X and Y. In other words, if Z contains two nonoverlapping substrings, one of each equals X and the other equals Y. If X contains Y or vice versa, substrings are considered overlapping.

Input

The first line of the input file contains the number of test cases (from 1 to 1000). The test cases are described in the following lines, three lines per each test case. The first line of test case contains representation of the signal Z. The following two lines of test case contain representations of the signals X and Y. Each of these strings is non-empty.

The total length of all lines in all test cases does not exceed 10^6.

Output

For each test case output "YES" (quotes for clarity) on a separate line if the signal Z is a combination of the signals X and Y, otherwise output "NO" (quotes for clarity).

Examples

input.txt	output.txt
3 abacada aba ada abacada aba aca abacada aba aaa	YES NO NO

Example explanation

In the second test case both strings X and Y are presented in the line Z as substrings. However, these substrings overlap: the third character of the string Z belongs to both of them.

Problem 13. Weather Balloon

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

Cowboy Joe has looked at the wind floating over the prairies for a long time. After these observations, he has created a formula that will describe the altitude of a weather balloon launched from his house. In particular, his equation predicts the altitude a (in metres above the ground) at hour t after launching his balloon is:

$$A = -6t^4 + ht^3 + 2t^2 + t$$

where h is an integer value representing the humidity as a value between 0 and 100 inclusive. Joe is curious at what the earliest hour is (if any) that his weather balloon will hit the ground after launch, so long as it is no more than the maximum time, M , that Joe is willing to wait. You can assume that the weather balloon touches ground when $A \leq 0$.

Input

The input is two non-negative integers: h, the humidity factor, followed by M — the maximum number of hours Joe will wait for the weather balloon to return to ground. You can assume $0 \leq h \leq 100$ and $0 < M < 240$.

Output

The output will be one of the following possibilities: "`The balloon does not touch ground in the given time.`" or "`The balloon first touches ground at hour:`", and, at new line, T — a positive integer, representing the earliest hour when the balloon has altitude less than or equal to zero.

Examples

input.txt	output.txt
30 10	The balloon first touches ground at hour: 6
70 10	The balloon does not touch ground in the given time.

Problem 14. Blood

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

At the hospital there are four types of blood available: O, A, B, and AB. Each of these types of blood has an Rh factor, which is either "positive" or "negative". There are many patients who each require 1 unit of blood. Each patient's blood type determines the type of blood he/she may receive:

- Each Type O patient requires Type O blood.
- Each Type A patient may receive blood of Type A or Type O.
- Each Type B patient may receive blood of Type B or Type O.
- The Type AB patients may receive blood of any type.

Patients who have Rh-negative blood can accept Rh-negative blood only, but patients with Rh-positive blood can accept either Rh-positive or Rh-negative types of blood. We want as many patients as possible to receive a unit of blood. What is the maximum number of patients that can receive blood?

Input

The first line of input contains 8 integers: the number of units of blood of Type O negative, O positive, A negative, A positive, B negative, B positive, AB negative and AB positive. This is followed by a line containing eight numbers: the number of patients whose blood type is O negative, O positive, A negative, A positive, B negative, B positive, and AB negative and AB positive. Each of these integers will be at least 0 and less than 10^7.

Output

The output of your program should be a single integer: the maximum number of patients that can receive blood.

Example

input.txt	output.txt
5 5 3 1 2 11 5 12 2 4 9 2 3 9 7 3	33

Problem 15. Card Game

Input file:	input.txt
Output file:	output.txt
Time limit:	6 seconds
Memory limit:	256 Mebibytes

Lets take is a two-player card game. Each card has a colour (Red, Yellow, Green, or Black) and a value (1, 2, 3, 4, or 5). The deck contains twenty cards; one card having each distinct combination of colour and value.

Each player is dealt ten of the twenty cards. The game is played in ten rounds, and the objective is to win as many rounds as possible. In each round, one player, the player with 'the lead', plays one of his cards. The other player must play a card of the same colour, if he has one. If not, he may play any of his cards. Both cards, involved in a round, are dropped out the game. The player with the lead (leader) wins the round if the other player has no card of the same colour, of if leader card has a higher value. Otherwise the other player wins the round.

The objective of the game is to win as many rounds as possible. The lead for the first round is chosen arbitrarily; the lead for each subsequent round is given to the winner of the previous round.

Your job is to determine how many rounds each player will win, assuming that each player employs the strategy that maximizes his advantage.

Input

The input contains T ($1 \le T \le 10$) test cases. Each test case consists of one line of input, identifying the cards dealt to player, who leads in first round. Each card is identified by a letter ('R','Y','G','B'), denoting its colour followed by a digit denoting its value (1,2,3,4,5). The other player receives the remaining cards in the deck. A line containing 10 stars, separated by spaces follows the last test case. This line should not be processed.

Output

For each test case, output a single line giving an integer between 0 and 10, the number of rounds won by player, who has the lead for the first round.

Example

input.txt	output.txt
G1 G3 B2 R2 Y1 R3 R5 Y2 Y3 G5 * * * * * * * * * *	3

Problem 16. Multiple choice tests

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

Mr. Oblomsky, teacher, likes to give multiple choice tests. One benefit of giving these tests is that they are easy to mark, given an answer key. The other benefit is that students believe they have a one-in-five chance of getting the correct answer, assuming the multiple choice possibilities are '`A`','`B`','`C`','`D`' or '`E`'.

Write a program that Mr. Oblomsky can use to grade one multiple choice test.

Input

The input will contain the integer N ($0 < N < 10^4$) followed by $2 \cdot N$ lines. These $2 \cdot N$ lines are composed of N lines of student responses (with one of '`A`', '`B`','`C`','`D`' or '`E`' on each line), followed by N lines of correct answers in same format and in the same order as the student answered the questions (that is, if line i is the student response, then line $N + i$ contains the correct answer to that question).

Output

Output the integer C which corresponds to the number of questions the student answered correctly.

Examples

input.txt	output.txt
3 A B C A C B	1
3 A A A A B A	2