icpc International Collegiate Programming Contest



The 2026 ICPC North America Southern California Regional Contest

Official Problem Set

Google Cloud
Google DeepMind
icpc diamond global sponsor

JETBRAINS
icpc global sponsor programming tools

Jane Street
icpc multi-regional titanium sponsor

north america platinum sponsor

north america institutional sponsor

2026/2027 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 1
FizzBuzz

FizzBuzz is a party game played by participants counting in turn. Starting at 1, each player names the next number in sequence. Every third number, the number is replaced with the word *Fizz*. Every fifth number, the number is replaced with the word *Buzz*. If these overlap, the number is replaced by *FizzBuzz*.

Your friends are playing FizzBuzz—they are very good at the game and do not make errors. You overheard N consecutive statements. However, you only started listening on some unknown turn T. Based on what you heard, what was the earliest possible turn T when you started listening?

The first line of input is one number N ($1 \leq N \leq 10{,}000$), the number of statements you heard. The next N lines are the consecutive statements that you heard. Each statement is either the string "`Fizz`", "`Buzz`", or "`FizzBuzz`", exactly as shown, or an integer in the range from 1 to 100,000 inclusive.

Print one integer: the earliest possible turn T when you started listening to the game.

Sample Input 1

```
5
Fizz
13
14
FizzBuzz
16
```

Output for Sample Input 1

```
12
```

Sample Input 2

```
1
Buzz
```

Output for Sample Input 2

```
5
```

2026/2027 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 2
Permutation Computation

You have 2 variables A and B, both of which are initialized to 1, and you are given some number of executable statements of the following 4 forms:

1. $A \mathrel{+}= A$
2. $A \mathrel{+}= B$
3. $B \mathrel{+}= A$
4. $B \mathrel{+}= B$

(In the statements above, "+=" is the addition-assignment operator, which adds the left-hand side and right-hand side together and stores the result in the left-hand side.)

Specifically, you are given o_1 copies of statement 1, o_2 copies of statement 2, o_3 copies of statement 3, and o_4 copies of statement 4.

You want to permute the statements in such a way that $\max(A, B)$ is maximized after executing all the statements in the permuted order. What is this value? Since this value is quite large, print out $\ln \max(A, B)$. Note that $\ln X$ represents the natural logarithm of X, which is the value v such that $e^v = X$.

Input to your program is four space separated integers o_1 o_2 o_3 o_4 ($0 \leq o_1, o_2, o_3, o_4 \leq 500$) on one line, representing the counts of statements 1, 2, 3, and 4. It is guaranteed that $1 \leq o_1 + o_2 + o_3 + o_4 \leq 500$.

Print a single number, representing the natural logarithm of the maximum value of either A or B across all permutations of the statements. Your answer will be accepted if it is within an absolute or relative difference of 10^{-6} from the judges' reference value.

Sample Input

```
1 2 1 1
```

Output for the Sample Input

```
2.8903717579
```

2026/2027 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 3
Lots of Infuriating Spurious Parentheses

You are coding in your favorite parenthetical language. The language consists of expressions, or sequences of parentheses, which are groups of "(" and ")" characters that are *balanced*. An expression B is *balanced* if it consists of zero or more balanced expressions concatenated together, or if it has the form (C), where C is a balanced expression.

You are typing a lot of these parenthetical expressions into your code editor, which comes with a fully functioning *linter*. The linter is a piece of software that aims to style and format your code neatly. For example, a very common linter will automatically indent lines of code to make it more readable. In this language, the linter knows how to remove redundant parentheses.

Redundant parentheses are the outer pair of parentheses when you have a (potentially empty) expression E that is wrapped by two pairs of consecutive parentheses: $((E))$. Redundant parentheses do not need to only be the first and last character. For example, the expression $()(())()$ has a pair of redundant parentheses.

Every pair of redundant parentheses will be removed by the linter. Some example expressions are shown after the linter has been applied.

- $(((()(())))) \rightarrow (()())$ after removing 2 pairs of parentheses
- $((((())))) \rightarrow ()$ after removing 3 pairs of parentheses
- $(()(()(()))) \rightarrow (()(()(())))$ after removing 0 pairs of parentheses

You have stumbled upon N parenthetical fragments, which are series of "(" and ")" characters that may or may not be balanced. You want to concatenate every ordered pair of fragments together (including a fragment with itself), and for every concatenation that forms a valid balanced expression, determine how many pairs of parenthesis the linter will remove.

The first line of input contains one integer N ($1 \leq N \leq 10^5$), representing the total number of fragments.

The next N lines of input contains a string s_i representing the i-th fragment. It is guaranteed that s_i only consists of "(" and ")" and the sum of characters does not exceed 10^6, that is to say $\sum_{i=1}^{N} |s_i| \leq 10^6$.

Let $r_{i,j}$ be the number of pairs of redundant parentheses that are removed when string i is concatenated to string j. Determine and print the sum of $r_{i,j}$ across all $1 \leq i, j \leq N$.

Sample Input 1

```
3
((
))
()))
```

Output for Sample Input 1

```
3
```

Problem 3
Lots of Infuriating Spurious Parentheses (continued)

Sample Input 2

```
2
(())
(())
```

Output for Sample Input 2

```
8
```

2026/2027 SOUTHERN CALIFORNIA REGIONAL
INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 4
Vickrey Auction

Swamp County is selling surplus items it no longer needs. The County traditionally auctioned surplus items using a *sealed-bid* auction, in which bidders submitted secret, sealed bids for an item. At the end of the auction, all bids were opened. The bids were ranked first by the price offered (high to low), and among bids of the same price, the time submitted (earliest to latest). The item was sold to the highest-ranked bidder for the price offered.

In a sealed-bid auction, bidders worry that they may be over-paying, so they tend to bid less than they might actually be willing to pay. Swamp County is now planning to use the "Vickrey auction" method instead. This is also a sealed-bid auction, with bids ranked as before. The difference is that the highest-ranked bidder gets the item, but pays the amount of the second-ranked bid. This auction style encourages bidders to offer what they see as the true value of the item.

As bids are submitted, bidders are assigned private bidder numbers in ascending order starting with 1.

Your team is write a program that determines the winner and price to be paid in a Vickrey auction. Input to your program begins with n, the number of bidders. n lines follow with the sealed bids of the buyers from 1 to n in order. There will be at least 2 and not more than 100 bidders. Each bid B is expressed in integer dollars, $1 \leq B \leq 1000$.

Your program is to print a line with two values separated by a single space: the number of the winning bidder, followed by the amount the bidder will pay.

Sample Input 1

```
2
4
10
```

Output for Sample Input 1

```
2 4
```

Sample Input 2

```
2
2
1
```

Output for Sample Input 2

```
1 1
```

Problem 4
Vickrey Auction (continued)

Sample Input 3

```
8
99
13
127
71
101
127
65
99
```

Output for Sample Input 3

```
3 127
```

2026/2027 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 5
King's Folly

A king has decided to reward a loyal subject with a grant of land. The subject may draw any axis-aligned polygon on a map, and the king will grant them all of the land inside it. Not wanting to seem greedy to the king, the loyal subject has drawn a map with limited borders.

Unfortunately, the king accidentally made the same promise to another loyal subject! The second loyal subject, also not wanting to appear greedy, drew a map with limited borders as well. The king has received both of the maps and wants to know how much land was promised twice. The two recipients will share the overlapping land, and the royal treasury must pay the loyal subjects an amount proportional to the area that was double-granted.

Write a program to help the king determine how much money his mistake will cost the royal treasury by calculating the area of land that was promised to both subjects.

Each polygon is specified by listing its distinct vertices in order around its boundary. Consecutive vertices are connected by an edge, and the last vertex is connected back to the first vertex. Every edge is horizontal or vertical, and consecutive edges meet at a 90° angle. Nonconsecutive edges do not touch. The vertices have integer coordinates.

In Figure 1, the left drawing shows two overlapping rectangular land grants. The right drawing shows that the grants might not be convex.

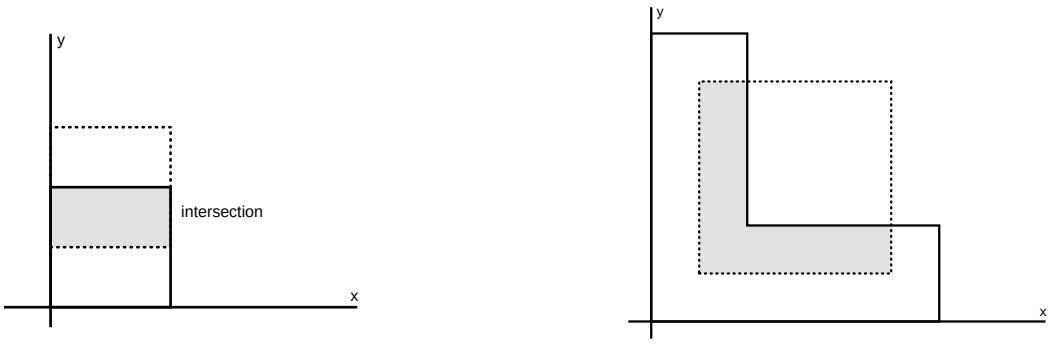


Figure 1. Drawings of the two examples in the Sample Input.

The first input line contains an integer N ($4 \le N \le 50{,}000$), the number of vertices in the first polygon. The next N lines contain the vertices of the first polygon in boundary order. Each line contains two integers x_i and y_i ($-1000 \le x_i, y_i \le 1000$). For every i from 1 through $N-1$, the edge from (x_i, y_i) to (x_{i+1}, y_{i+1}) is horizontal or vertical. The edge from the last vertex back to the first vertex is also horizontal or vertical. Consecutive edges meet at a 90° angle.

The next line contains an integer M ($4 \le M \le 50{,}000$), the number of vertices in the second polygon. The next M lines describe the second polygon in the same format.

Your program is to print one integer: the area of the intersection of the two polygons.

Problem 5
King's Folly (continued)

Sample Input 1

```
4
0 0
0 2
2 2
2 0
4
0 1
0 3
2 3
2 1
```

Output for Sample Input 1

```
2
```

Sample Input 2

```
6
0 0
0 6
2 6
2 2
6 2
6 0
4
1 1
1 5
5 5
5 1
```

Output for Sample Input 2

```
7
```

2026/2027 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 6
Probabilistic Descent

We are given an integer n ($1 < n \leq 18$) and a perfect binary tree of depth $n-1$. The binary tree has exactly $2^n - 1$ vertices (so each leaf has depth $n-1$), the root has number one and each vertex i has left child $2i$ and right child $2i+1$. In each vertex, there is an integer x_i, $0 < x_i < 2^n$ (except $x_1 = 0$).

We are going to play a game on the given binary tree. The object of the game is to maximize the total score by descending the tree one or more times. The total score starts at zero. Points start at zero and are accumulated by making moves that descend the tree.

- In order to move, a random integer r is drawn between 0 and $2^n - 1$ inclusive. The player can move to any child where $x_i \wedge r = 0$. ($\wedge$ is the bit-wise AND operator.) Moving to child i adds x_i points to the descent.
- Before each move, the player can choose to end the game instead and add the points accumulated from the current descent to the total score.
- Should the player attempt to move, but the random integer drawn is such that neither child can be reached, the game ends immediately. No points are received for the current descent and the total score is final.
- If one of the tree's leaf nodes is reached, all the accumulated points for the descent are added to the total score. The player can end the game or start another descent back at the root with points reset to zero.

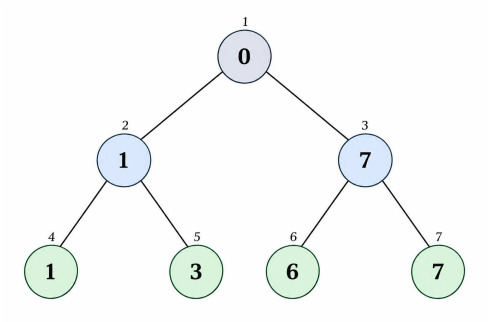


Figure 1. The binary tree from Sample Input 2.

Assuming optimal plays are made, what is the expected ending score?

The first line of input contains the integer n, $1 < n \leq 18$. The second line of input has $2^n - 1$ integers, separated by spaces, where the ith integer is x_i.

Output the expected ending score if you play the game optimally. Your answer will be accepted if the relative or absolute difference from the judges' reference value does not exceed 10^{-5}.

Sample Input 1

```
2
0 1 2
```

Output for Sample Input 1

```
5
```

Problem 6
Probabilistic Descent (continued)

Sample Input 2

```
3
0 1 7 1 3 6 7
```

Output for Sample Input 2

```
1.7692307694
```

Sample Input 3

```
4
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Output for Sample Input 3

```
0.5
```

2026/2027 SOUTHERN CALIFORNIA REGIONAL
INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 7
Memory Game

This is an interactive problem.

You are going to play the "Memory Game". The rules are simple. There are n matching pairs of cards. The face sides of both cards in a pair show the same image, while each pair's image is unique. The cards are placed face down, shuffled, and laid flat so they can be picked up individually. The cards are numbered from 1 to $2n$ inclusive. In each turn you can pick up any two cards (one at a time, so knowing what you see on the first card can affect which card you will pick up as the second one). If they match, you take them off the board. If not, you return them to their original places, and put them face down again. The card numbers do not change during the game. Your team is to write a program that will match all the cards, using at most $2n - 1$ moves.

Your program is to begin by reading a line with n, the number of card pairs ($2 \leq n \leq 100$). Your program is to then play the game by making moves. Each move consists of picking up two cards. To pick up the first card, output a line with the number of the card your program is selecting. After you print the first number, your program will receive on standard input the name of the image on the selected card, as a string of 1 to 10 upper-case letters. Each image has a unique name. Output another line with the number of the second card your program is selecting. Read the name of the second image on standard input. Repeat making moves until you believe you have won the game by removing all the pairs. At that point, you should output the string "`VICTORY`" on a single line and end your program.

Your program is to track removed pairs and output only valid moves. If you output an invalid move (picking up the same card twice in a single move, picking up a card that has already been removed, or picking up a card with an invalid number), you will immediately lose. If you use more than $2n - 1$ moves, declare victory too early, or forget to declare it at all, you will lose as well.

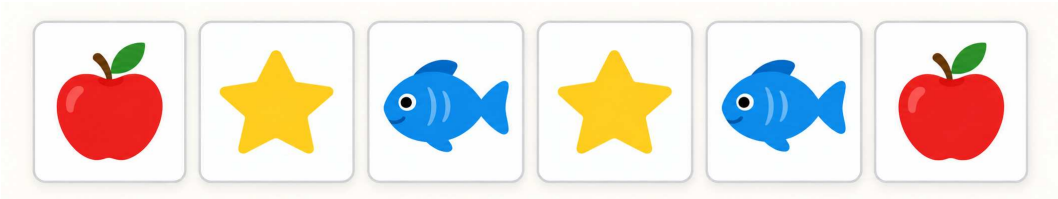


Figure 1. Memory Game cards corresponding to the Sample Interaction.

Problem 7
Memory Game (continued)

Sample Interaction

Read	*Write*
3	
	2
STAR	
	3
FISH	
	1
APPLE	
	6
APPLE	
	4
STAR	
	5
FISH	
	2
STAR	
	4
STAR	
	3
FISH	
	5
FISH	
	VICTORY

Problem 7
Memory Game (still continued)

Judging Notes on Interactive Problems

- In most programming environments, program output is buffered to speed up I/O operations. With interactive problems, it is crucial to make sure the output is actually sent from your program and not simply stored in internal buffers. This typically means flushing the output buffers after each line of output (that is, after each newline character) as follows:
  - In C (or C++ using *cstdio*), you can use `fflush(stdout)`.
  - A C++ output stream is flushed automatically each time you write the `endl` manipulator. When using other means or if you want to be sure, call `cout.flush()`.
  - In Java and Kotlin, the `System.out` stream has so-called "auto-flush" functionality and its buffer is therefore flushed automatically with each newline character. When using other streams or if you want to be sure, invoke the `flush()` method of the stream.
  - In Python, you can use `sys.stdout.flush()`.
- Interactive problems are judged in a way similar to other problems, but there are some differences:
  - When your program attempts to read data, it will wait until more data are available or until the judge program terminates the input (unless you read in a non-blocking way, which is beyond the scope of these notes). Thus, if your program attempts to read more input than can currently be provided (e. g., because you forgot to flush your previous output, or because of some other reason), then the program will stall indefinitely and your submission will get Time Limit Exceeded.
  - As usual, the judgment given to an incorrect submission is the first error discovered, but this does not always mean exactly the same thing as for traditional problems. For instance, if your submission to a traditional problem is too slow on a particular test case, you would get Time Limit Exceeded on that test case. With an interactive problem, the judgment may be Wrong Answer if the solution exhibits an incorrect answer before it runs out of time.
  - The time limit for an interactive problem is how much time your submission may spend; the time spent by the judge program is not counted towards this.
  - Because of timing between the interactive judge program and your submission, judgments for incorrect submissions to interactive problems are not necessarily deterministic. We can guarantee the following: a judgment of Wrong Answer means that your submission produced incorrect output, and a judgment of Run Time Error means that your submission returned a non-zero exit code. If your submission does both, you might receive either judgment.

There is a sample interactor testing tool available as a problem attachment. To test your solution:
`python3 sample_interactor.py` *your-solution* [`--verbose`]

For more information, run:
`python3 sample_interactor.py --help`

2026/2027 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 8
Gossips

Your classmates love gossiping. Each one of them has their favorite gossip buddy, to whom they tell everything they have heard. The problem is, their buddy will immediately run and tell *their* gossip buddy (who cares that it is all "Top Secret, do not tell *anyone*."). This leads to a risk where you tell your buddy a secret piece of gossip, one that only you know, just to hear it a few days later from someone other than your own gossip buddy. This makes you feel utterly betrayed. Your classmates feel the same way if and when this happens to them.

Knowing your classmates and their gossip buddies, you are to write a program to identify who is at risk of feeling utterly betrayed one day.

The first line of input is the number of classmates as an integer n ($2 \leq n \leq 10^5$). The next n lines describe your classmates. Each line contains two strings consisting of at least one and at most five uppercase English letters (A–Z), separated by a single space. The first string is the name of your classmate, and the second the name of their gossip buddy. You are guaranteed that all the names of your classmates are unique. You are also guaranteed that any person named as a gossip buddy will appear in the input as your classmate (and therefore you know their gossip buddy). No classmate has themselves as a gossip buddy.

Print a line with a number x, the number of classmates at risk of feeling utterly betrayed. Follow that with x lines giving the names of these classmates in alphabetical order.

Sample Input 1

```
4
TOM JERRY
JERRY TOM
ADAM TOM
ANGIE JERRY
```

Output for Sample Input 1

```
0
```

Sample Input 2

```
3
LAURA JACOB
JACOB ANDY
ANDY LAURA
```

Output for Sample Input 2

```
3
ANDY
JACOB
LAURA
```

2026/2027 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 9
TriStar

A figurate number is a number that can be displayed by arranging dots on a grid to make a geometric pattern.

Triangular numbers, as shown in Figure 1, are an example of figurate numbers. They form a nice sequence:

$$1, 3, 6, 10, 15, 21, 28, \ldots$$

Another, more interesting example, are star numbers as shown in Figure 2. They form another sequence:

$$1, 13, 37, 73, 121, 181, 253, \ldots$$

Star and triangular numbers are related. A star consists of six points. Each point of the star consists of two triangular figures, with each figure composed of the same triangular number of dots. Triangular figures for two points are shown in the 37-dot star of Figure 2. The triangular figures are composed of 3 dots each.

There are also infinitely many numbers that are both a triangular number and a star number. That sequence:

$$1, 253, 49141, 9533161, \ldots$$

Your team's job is to write a program that will take a list of numbers and report whether each number is triangular, star, both, or neither.

Input to your program is a series of lines terminated by end of file. Each line consists of one to twenty positive integers, n, separated by white space. Each integer n: $1 \leq n < 2^{61} - 1$. No input line is longer than 100 characters. There will be at least 1 and no more than 200 integers in the input.

For each integer, print a line with n, a single space, and one of the following results:

- "`t`" if the number is triangular only,
- "`s`" if the number is star only,
- "`ts`" if the number is both triangular and star, or
- "`n`" if the number is neither.

Sample Input

```
1   3 6 10 15
    13 37    73 121 181
   253 49141  9533161   1849384153
2305843009213693950 13501987525271953
660721321 523 83 536887296 36877296
```

Problem 9
TriStar (continued)

Output for the Sample Input

```
1 ts
3 t
6 t
10 t
15 t
13 s
37 s
73 s
121 s
181 s
253 ts
49141 ts
9533161 ts
1849384153 ts
2305843009213693950 n
13501987525271953 ts
660721321 n
523 n
83 n
536887296 t
36877296 n
```

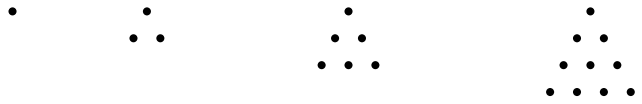


Figure 1. Triangular Numbers.

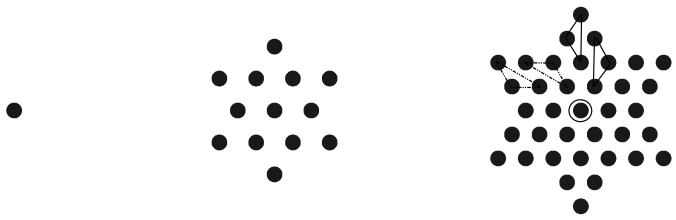


Figure 2. Star Numbers.

2026/2027 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 10
Mathematical Card Trick

You, a novice magician, are practicing a mathematical card trick with your friend, who will be your assistant.

The trick is performed with a standard deck of 52 playing cards. A deck is divided into four distinct suits: Clubs (♣), Diamonds (♢), Hearts (♡), and Spades (♠). Each suit contains 13 cards of varying ranks. Low to high, the ranks are: 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King, and Ace.

The trick goes as follows:

1. The assistant invites a spectator to pick 5 cards from the deck.
2. The assistant hides one of the cards (say, in the spectator's pocket) and shows the other 4 cards to the magician.
3. The magician correctly identifies the identity of the 5th card just by looking at the 4 cards shown.

To make this trick possible, you and your friend decide to use the following secret protocol.

Because the spectator chooses 5 cards, at least 2 of the cards must share the same suit. If there are multiple cards of more than one suit, the assistant chooses an arbitrary suit. Call the chosen suit S.

Consider all 13 cards of suit S and imagine arranging them around a circle in clockwise order, so that the 2 is the clockwise neighbor of the Ace. Let the ranks of two cards of suit S chosen by the spectator be x and y. Since there are 13 ranks in a suit, the shortest distance between x and y measured along the circle must be at most 6. The choice of which card is x and which y is made so that y is at most 6 clockwise ranks away from x. If there are multiple such pairs of cards, the assistant may choose any two for x and y. Let the clockwise distance from x to y be d.

For example, Figure 1 shows that the 3 of Spades is 5 clockwise ranks away from the Jack of Spades. For this pair of cards, x is the Jack of Spades, y is the 3 of Spades, and d is 5.

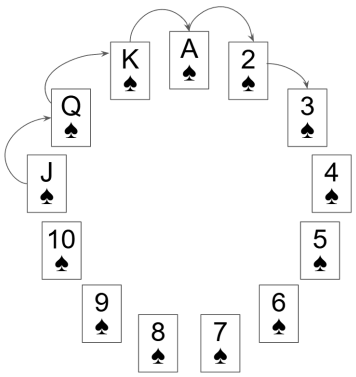


Figure 1. The 3 of Spades is 5 ranks away from the Jack of Spades.

The assistant then hides the card of suit S and rank y. The remaining 4 cards are arranged so that the card with suit S and rank x is shown first, and the remaining 3 cards encode the distance d via their permutation. We predetermine the ordering of the cards as follows:

- Use the rank first to determine the ordering: $2 < 3 < \ldots < 10 <$ Jack $<$ Queen $<$ King $<$ Ace.
- Within the same rank, use the suit: Clubs $<$ Diamonds $<$ Hearts $<$ Spades.

With this ordering, the 2 of Clubs is the "smallest" card, and the 2 of Diamonds is the second smallest. The Ace of Spades is the "largest" card.

Problem 10
Mathematical Card Trick (continued)

Let the 3 cards be a, b, and c, where $a < b < c$ according to the ordering above. The permutation of these cards determines d:

- abc: $d = 1$
- acb: $d = 2$
- bac: $d = 3$
- bca: $d = 4$
- cab: $d = 5$
- cba: $d = 6$

The magician then sees the 4 cards. From the first card, the magician infers the suit S and the starting rank x. Then, from the next 3 cards, the magician infers the distance d, computes the rank y (wrapping around the ranks as needed), and declares the hidden card to be the one with suit S and rank y!

Since the trick requires both you and your friend to be deeply familiar with the protocol, you decide to write a program to train yourselves.

Input starts with T ($1 \leq T \leq 10{,}000$), the number of test cases. Each test case is given on a single line. When a line starts with the uppercase letter "`A`", it is followed by 5 space-separated strings representing the 5 cards chosen by the spectator. Otherwise, the line starts with the uppercase letter "`M`", and it is followed by 4 space-separated strings representing the 4 cards shown to the magician by the assistant.

Each card is specified by its rank first (`2`, `3`, ..., `9`, `T`, `J`, `Q`, `K`, `A`, where `T` denotes 10, `J` denotes Jack, `Q` denotes Queen, `K` denotes King, and `A` denotes Ace), followed by its suit (`C` for Clubs, `D` for Diamonds, `H` for Hearts, and `S` for Spades). For example, `KD` represents the King of Diamonds, `TS` represents the 10 of Spades, and `2C` represents the 2 of Clubs.

For each input line starting with the uppercase letter "`A`", print the 4 cards that must be shown to the magician in order, according to the secret protocol. If there are multiple possible solutions, print any of them.

For each input line starting with the uppercase letter "`M`", print a string that represents the hidden card. You may assume that the 4 cards are correctly presented according to the secret protocol.

Sample Input

```
6
M 2S JS TH 6C
M 9C 5S 5H 8D
A 6S KH 5C AD 3C
A 7H QH 7C 4H 3D
M TS QS 4D KD
A TD 9D QD 2H 4C
```

Output for the Sample Input

```
8S
QC
3C 6S AD KH
4H 7C 3D QH
KS
9D 2H 4C QD
```

2026/2027 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 11
Sticker Scraping

You and your friend are sitting at a table with a triangle-shaped sticker in the middle. You have sticker scrapers, and you are taking turns scraping up the sticker. Because you are already sitting, you only can only scrape from a fixed direction, and you can only scrape a limited amount of the sticker at a time, as described below. You and your friend really like scraping, so now you wonder: what is the maximum number of times you can scrape the sticker until it is gone?

For this problem, we will consider the scrapers to be infinite lines that are swept partway across the plane, intersecting part of the sticker. When scraping, you do the following process:

1. Move your scraper forward until it intersects a vertex of the unscraped portion of the sticker.
2. Continue moving your scraper forward a nonzero distance until it touches another vertex of the unscraped portion of the sticker.
3. Remove the portion of the sticker that your scraper passed through, leaving behind any unscraped portion of the sticker.

You and your friend may scrape the sticker in any order.

Given the coordinates of the triangle and the directions of your scrapers, compute the maximum number of times you and your friend can scrape the sticker, or if you can continue scraping the sticker forever.

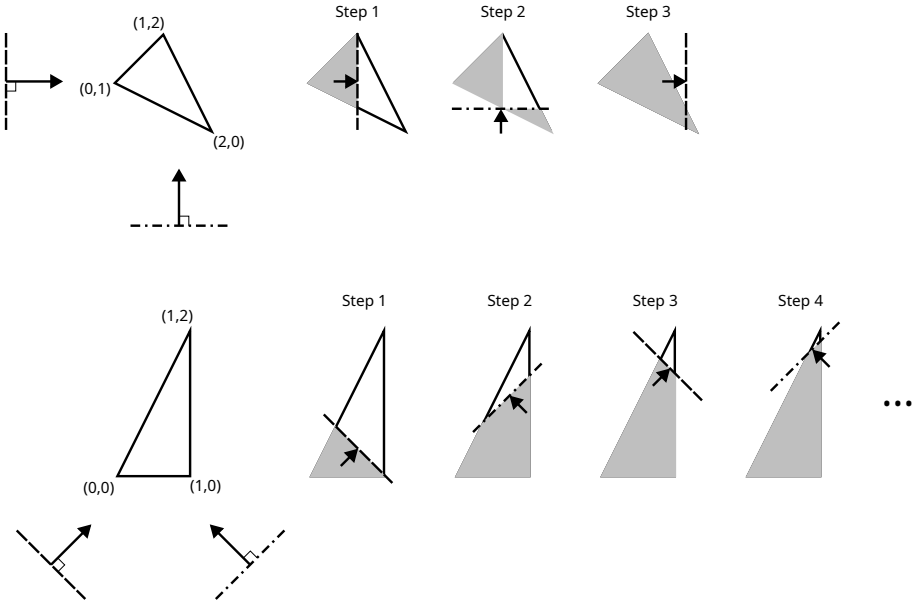


Figure 1. Sticker scraping diagrams corresponding to the Sample Input.

In Figure 1 above, the top sticker can be scraped 3 times. The bottom sticker can be scraped infinitely many times.

The first line of input is an integer N, the number of test cases: $1 \leq N \leq 1000$. The next N lines contain the test cases. On each line, there are ten space-separated integers x_1, y_1, x_2, y_2, x_3, y_3, dx_1, dy_1, dx_2, and dy_2.

Each pair (x_i, y_i) denotes the (x, y) coordinates of the ith sticker vertex: $|x_i| \leq 100$, $|y_i| \leq 100$. The sticker is guaranteed to have a non-zero area.

Problem 11
Sticker Scraping (continued)

Each pair (dx_i, dy_i) denotes the scraping direction of the ith scraper. When scraping, the scraper starts far away from the triangle and is translated along the direction (dx_i, dy_i): $|dx_i| \leq 100$, $|dy_i| \leq 100$. It is guaranteed that dx_i and dy_i are not both zero.

Print N lines, the answer to each test case in order. For each test case, print the maximum number of times you can scrape the sticker, or the string "`forever`" if you can continue scraping forever.

Sample Input

```
2
0 1 1 2 2 0 1 0 0 1
0 0 1 2 1 0 1 1 -1 1
```

Output for the Sample Input

```
3
forever
```

2026/2027 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 12
Apartment Hunting

You are hunting for a new apartment, and you need to schedule tours. The rental market is very competitive right now, so even if you tour an apartment and put in an application, there is a good chance you will not get it. Plus, many apartment tours are at the same time, and you cannot be in two places at once.

You have a list of N apartments you want to tour. For the ith apartment there is exactly one tour, which occurs in the timespan $(start_i, end_i)$. In order to put in an application and have a chance at getting the apartment, you must attend the tour for the entire timespan. You cannot tour another apartment whose timespan overlaps for a non-zero duration. However, the apartments you are touring are all very close to each other, so it takes negligible time to get from one tour to another.

Your credit and references are already good, so you only need to worry about whether your application will be approved. Fortunately, you have a really good intuition. If you apply for apartment i, you know that with probability p_i your application will be approved and you will get the apartment. The chance of getting one apartment is independent of the chance of getting any other apartment. It is okay to get more than one apartment; in that case, you will simply withdraw your extra applications.

Assuming you plan your schedule optimally, compute the probability that you will get at least one apartment.

The first line of input is an integer N, the number of apartments: $1 \leq N \leq 10^5$. Each of the next N lines is a description of the tour for each apartment. The ith line contains space-separated values $start_i$, end_i, and p_i, where $start_i$ and end_i are the integer timespan endpoints $0 \leq start_i < end_i \leq 10^9$, and p_i is a real number denoting the probability of getting the ith apartment, $0 \leq p_i \leq 1$, specified with up to 6 digits after the decimal point.

Output a line containing a single real number: the probability of getting at least one apartment with an optimal schedule. Your answer will be accepted if the relative difference compared to the judges' reference answer does not exceed 10^{-6}.

Sample Input 1

```
4
2 4 0.5
5 7 0.75
2 6 0.25
9 10 0.25
```

Output for Sample Input 1

```
0.90625
```

Problem 12
Apartment Hunting (continued)

Sample Input 2

```
3
2 4 0.5
5 7 0.75
3 6 1.0
```

Output for Sample Input 2

```
1.0
```

Sample Input 3

```
3
1 5 0.5
4 5 0.6
5 7 0.1
```

Output for Sample Input 3

```
0.64
```