Problem A. Closest Restaurant

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	256 megabytes

Imagine wandering through the streets of a bustling city, your stomach growling louder than the traffic around you. In this digital age, where a smartphone can pinpoint many dining options within seconds, the quest for the nearest culinary delight has transformed from a daunting challenge into a thrilling adventure.

Given a list of queries on a 2D plane, where each query either announces the grand opening of a new restaurant with its geographical coordinates or emanates from a city citizen asking where to find the nearest restaurant to their current location, your challenge is to process these queries.

Each restaurant will claim its unique spot on the map, as do the queries from potential clients. Consider the city an immense grid where distances between points are measured as their Euclidean distance.

Input

The first line contains an integer Q, the number of queries ($1 \le Q \le 2 \cdot 10^5$).

The next Q lines describe the queries. Each query is either of the form:

- 1 name x y : to add a restaurant named name ($5 \le |name| \le 20$) at coordinates (x, y), where name is a unique string consisting of lowercase English letters, and x and y are integers ($0 \le x, y \le 10^9$).
- 2 x y : to find the nearest restaurant to the point (x, y), where x and y are integers ($0 \le x, y \le 10^9$).

Output

For each query of the second type, output a single line containing the name of the nearest restaurant. If there are multiple answers, print the lexicographically smallest name.

Example

standard input	standard output
5 1 lepizzaria 1 2 1 papajohns 3 4 1 tacotown 0 1 2 0 0 2 3 3	tacotown papajohns

Note

you may assume that the restaurants are uniformly distributed across the city, ensuring no area is a culinary desert or overly saturated with options.

Problem B. Double Gamble

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

Anas and Rida have each collected a set of numbered balls. Anas's set contains n balls, while Rida's set has m balls. The twist is that neither Anas nor Rida knows the numbers written on their balls until they select them.

They decide to play a game using their sets under these conditions. Both Anas and Rida will simultaneously select three random balls from their respective sets. Only after they have made their selections will they reveal the numbers on their chosen balls and note down the sum of these numbers. The person with the higher sum wins the game. If both have the same sum, the game ends in a tie.

Your task is to determine the probability that Anas wins the game, considering the added layer of uncertainty about the numbers on the balls until they are chosen.

Input

The first line contains two integers n and m ($3 \le n, m \le 10^5$), the number of balls in Anas's and Rida's sets, respectively.

The second line contains n distinct integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i < 2^{16}$), the numbers on Anas's balls.

The third line contains m distinct integers $b_1, b_2, \ldots, b_m$ ($1 \le b_i < 2^{16}$), the numbers on Rida's balls.

Output

Print a single real number — the probability that Anas wins the game. Output it with a fixed precision of 10^{-6}

Examples

standard input	standard output

`3 3`
`1 2 1`
`1 1 1`
	`1.000000`

`4 5`
`4 4 4 5`
`2 5 8 3 4`
	`0.375000`

Problem C. Traveler's Dilemma

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

Anas is a traveler in the realm of Vastland and is planning his journey there. Vastland consists of N ($1 \le N \le 10^5$) cities connected by M ($1 \le M \le min(N \cdot (N-1), 2 \cdot 10^5)$ one-way roads. Each city is uniquely numbered from 1 to N, and every city has an airport. The roads are directed, leading from one city to another.

Anas wants to visit as many cities as possible and has asked for your help. His objective is to explore Vastland in such a way that he can visit the maximum number of cities. He will begin his journey from the airport of any city. Then after his airplane lands, He can only travel along the one-way roads and he may use the same road multiple times if it helps in exploring more cities. The journey ends when he cannot reach any more cities.

Given the structure of Vastland, determine the maximum number of cities Anas can visit in a single journey if he selects his starting point optimally.

Input

The first line contains two integers N and M ($1 \le N \le 10^5, 1 \le M \le min(N \cdot (N-1), 2 \cdot 10^5)$ — the number of cities and one-way roads, respectively.

Each of the next M lines contains two integers a_i and b_i ($1 \le a_i, b_i \le N$), indicating that there is a road from city a_i to city b_i. It's guaranteed that no road connects a city to itself, and there are no multiple roads connecting the same pair of cities in the same direction.

Output

Print a single integer — the maximum number of cities that can be visited in one journey if Anas chooses his starting city optimally.

Examples

standard input	standard output

4 4
1 4
2 1
3 2
4 1
	4

5 5
3 2
5 1
1 4
3 5
2 4
	4

Note

In the first example, Anas can start his journey from city 3, then travel to cities 2, 1, and 4, visiting all the cities in Vastland.

Problem D. Another Fibonacci Problem

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

Given a very large integer N where $0 \leq |N| \leq 10^5$, your task is to find the n-th Fibonacci number modulo 37.

The Fibonacci sequence is defined as follows: $F_0 = 0$, $F_1 = 1$, and for each $i \geq 2$, $F_i = F_{i-1} + F_{i-2}$.

Input

The first line of the input contains a single integer T ($1 \leq T \leq 10^4$), the number of test cases.

Each of the next T lines contains a single very large integer N ($0 \leq |N| \leq 10^5$), where $|N|$ is the number of digits in N.

It is guaranteed that the sum of the lengths of all numbers in all the test cases does not exceed 4×10^5.

Output

For each test case, output a single line containing one integer: the n-th Fibonacci number modulo 37.

Example

standard input	standard output
3 3 8 12	2 21 33

Problem E. Wasted Librarian

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

Ahmed, a librarian with a passion for organizing his library's books, sees the arrangement as a key to simplifying life. As Ahmed's friend and an exceptional computer scientist, you've taken it upon yourself to help in maintaining the precision of his book organization system.

To do this, Ahmed will present you with a series of queries regarding the library's inventory. Occasionally, he might request you to figure out how many books have titles that alphabetically fall between two specific strings, a method that ensures his organization system's accuracy.

Your task is to process N queries, which can be of two types:

- 1. Register copies of the book in the library's inventory.
- 2. Determine the number of books whose titles are alphabetically between two given strings.

Input

The first line of the input contains a single integer N ($1 \le N \le 10^5$), the number of queries.

Each of the next N lines describes a query and is of one of the following forms:

- 1 *name* c: Register c copies of book with title *name* in the library's inventory, where *name* is a string consisting of lowercase alphabet letters ($1 \le |name| \le 20$), and c is an integer ($1 \le c \le 10^6$).
- 2 *low* *high*: Determine the number of books whose titles are alphabetically between *low* and *high*, where *low* and *high* are strings consisting of lowercase alphabet letters ($1 \le |low|, |high| \le 20$), representing the alphabetical range.

Output

For each Type 2 query, output a single line containing an integer, the number of books whose titles are alphabetically between *low* and *high*, inclusive.

Example

standard input	standard output
8 1 cat 1 1 dog 2 1 cream 1 1 apple 3 1 test 2 2 apple cat 2 app cattle 2 arc zoo	4 4 6

Note

Note that Ahmed may register copies of the same book more than once.

Problem F. Even Odd Country

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

Consider a country that contains N cities ($1 \leq N \leq 10^5$). Each city is assigned an integer value x ($1 \leq x \leq 10^9$). The cities are connected through a series of bidirectional roads, forming a network.

You will be provided with a list of these roads, each indicating a direct connection between two cities. Alongside the network, you will receive Q queries ($1 \leq Q \leq 2 \times 10^5$), where each query consists of two city identifiers, a and b ($1 \leq a, b \leq N$).

The task for each query is to determine whether there exists a path between city a and city b such that all cities along the path have the same parity of their associated values. Here, parity refers to the oddness or evenness of the city's value.

Input

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

The next line contains N integers, where the i-th integer represents the value associated with the i-th city.

The following line contains the integer M ($1 \leq M \leq 2 * 10^5$), the number of roads.

Each of the next M lines contains two integers u and v ($1 \leq u, v \leq N$), indicating a road between cities u and v.

The next line contains the integer Q ($1 \leq Q \leq 10^5$), the number of queries.

Each of the following Q lines contains two integers a and b ($1 \leq a, b \leq N$), representing a query to check whether there is a path between city a and city b consisting entirely of cities with the same parity.

Output

For each query, print 'YES' if there is a path between city a and city b consisting entirely of cities whose values have the same parity. Otherwise, print 'NO'.

Example

standard input	standard output

5 5
2 4 6 1 7
1 2
2 3
3 5
1 4
4 5
3
1 3
2 4
4 5
	YES
	NO
	YES

Problem G. Unexpected Behaviour

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

Consider the following grammar:

- `<statement> ::= <assignment> | <retrieval>`
- `<assignment> ::= set <identifier> = <expression>`
- `<retrieval> ::= get <identifier>`
- `<expression> ::= <term> | <term> + <expression> | <number> '*' <expression>`
- `<term> ::= <identifier> | <literal> | '(' <expression> ')'`
- `<identifier> ::= [A-Za-z]+`
- `<literal> ::= ".*"`
- `<number> ::= [1-9]`

This grammar describes a string using the following rules:

`<literal>` describes itself (without the quotes), `<identifier>` describes the value stored by the identifier if already set, otherwise consider it as a literal value.

`<number> * <expression>` describes the string described by `<expression>` repeated `<number>` times.

`<expression> + <term>` describes the concatenation of strings described by `<expression>` and `<term>`.

the statement `set <identifier> = <expression>` evaluates `<expression>` and stores its value in `<identifier>`. If `<expression>` contains identifiers that have not been set previously, these identifiers are treated as literal values.

the statement `get <identifier>` outputs the latest value stored in `<identifier>`. if the identifier have not been set previously, the identifier will be treated as a literal value.

For example `2*"a"+(c+3*("CAT"+b))` describes the string `"aacCATbCATbCATb"`.

In the example above, the variables `c` and `b` are not set; hence, they are treated as literal values.

Input

The first line contains a single integer n $(1 \le n \le 10^4)$ — the number of statements to evaluate.

Each of the next n lines describes a single operation. An operation is either an assignment or a retrieval, formatted as in either formats:

- `set <identifier> = <expression>`
- `get <identifier>`

Where `<identifier>` is a non-empty string consisting of uppercase and lowercase English letters and `<expression>` can be any valid expression based on the grammar rules provided. The total length of all expressions `<expression>` in each assignment will not exceed $4 * 10^5$ characters. The total length of the evaluations of all the `<expression>` will not exceed $4 * 10^6$

Output

For each `get` operation, output a single line containing the value of `<identifier>`. If `<identifier>` has not been set before the retrieval, output its literal value.

Example

standard input	standard output
`10`	`ice`
`set a = "ice"`	`cream`
`get a`	`icecream`
`set b = "cream"`	`yoyo`
`get b`	`bonbon`
`set c = a + b`	
`get c`	
`set d = 2 * "yo"`	
`get d`	
`set e = 2 * (bo + "n")`	
`get e`	

Note

It is guaranteed that each `expression` will not be using unnecessary nested parentheses.

Problem H. Wedding Planner

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

Your friend, in the midst of wedding preparations, has encountered a dilemma and has turned to you for assistance. He wishes to invite exactly N $(1 \le N \le 4 \cdot 10^3)$ guests to his wedding. As you begin to help him draft the guest list, it becomes apparent that there are conflicts among some of the potential guests: certain individuals have expressed that they would prefer not to attend if certain others are also present.

Determined to ensure that his wedding is a joyful event for every guest, you've proposed a solution: divide the wedding into two separate celebrations, ensuring that no guest is required to share space with someone with whom they have a conflict. Your task now is to organize your guests into these two celebrations in such a way that everyone is comfortable, while also aiming to minimize the difference in the number of guests between the two celebrations.

Given the number of guests N and the list of conflicts of length M, determine whether it's possible to arrange the wedding celebrations as planned. If a satisfactory arrangement can be made, you have to find the minimum possible difference in the number of guests between the two celebrations.

Input

The first line contains two integers N and M $(1 \le N \le 4 \cdot 10^3,\ 0 \le M \le \min\big(N \cdot (N-1), 2 \times 10^5\big))$ — the number of invitees and the number of conflicts, respectively.

Output

Print a single integer — the minimum difference in the number of people between the two sections if the party can be split according to the conditions. If it's impossible, print -1.

Examples

standard input	standard output

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

`5 5`
`1 3`
`4 5`
`3 2`
`1 4`
`2 1`
	`-1`

Note

In the first example, the party can be split as follows (1, 3) and (2, 4), thus making the difference 0.

In the second example, it can be proven that it's impossible to split the party as required.

Problem I. Olive Farm

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

Anas is preparing to plant N olive trees on his farm to produce olive oil. His field is represented as a 1D number line, and he has identified M $(1 \le M \le 10^5)$ specific, non-overlapping intervals marked as fertile ground suitable for the trees. Anas aims to plant each of the N trees $(2 \le N \le 10^5)$ at distinct positions within these intervals to maximize the spacing D. The spacing D represents the minimum distance between any two trees, ensuring each tree receives healthy growth and maximum sunlight exposure.

Input

The first line contains two integers N and M, where N is the number of olive trees to plant and M is the number of fertile ground intervals.

Each of the next M lines describes an interval with two integers a and b, where $(0 \le a \le b \le 10^{18})$. No two intervals overlap or touch at their endpoints. An olive tree planted on the endpoint of an interval counts as standing on fertile ground.

Output

Output the largest possible value of D such that all pairs of olive trees are at least D units apart. It is guaranteed that a solution with $D > 0$ exists.

Example

standard input	standard output

`4 3`
`0 4`
`6 7`
`11 14`
	`3`

Problem J. Said Logs

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

Said, a software architect, begins his day by analyzing the logs generated by his company's server to ensure the systems are operating smoothly. He looks for specific patterns in the logs, which he refers to as *fortunate*. These patterns are essential for confirming successful and error-free operations.

A *fortunate* pattern is defined as a sequence in the log files that starts with the line `ERROR`, followed by any number of lines, and ends with the line `RESOLVED`. This sequence indicates an error that was successfully identified and resolved during server operations.

Given the content of a log file, determine the count of unique "resolved errors" within the logs. It is important to note that a `RESOLVED` line can only be used to resolve one `ERROR`.

Input

The first line of the input contains a single integer T ($1 \leq T \leq 8 * 10^3$), the number of logs in the log file.

After that, Each of the next N lines contains a string S ($4 <= |S| <= 20$) where S is the type of the log.

Output

Output one number, the number of resolved errors in the log file.

Example

standard input	standard output
5 ERROR RESOLVED ERROR ERROR RESOLVED	2