IEEEXTREME PROGRAMMING COMPETITION 15.0

Table of Contents

Welcome to IEEEXtreme 15.0

Time limit: *1280 ms*
Memory limit: *264 MB*

On behalf of the executive committee, Welcome to the 15th IEEEXtreme Programming contest! A full year has passed and here we meet again for another great programming party! Hundreds of volunteers have been working for months to prepare this event and we are glad to have you all aboard.

Our slogan for IEEEXtreme 15.0 is **Contemplate, Code, and Conquer**, which signifies your chance to come in, contemplate, code and conquer your place amongst more than 12\,00012000 of your peers from 6262 countries across the globe.

As implied by its name, IEEEXtreme can truly be “extreme”. Be sure to get plenty of rest, stay hydrated, and don’t forget to eat. Above all, have as much fun as possible! Getting a good standing in the final scoreboard will find you with great prizes and endless bragging rights; however, ensuring that you have the most enjoyable experience, will leave you with lifelong memories!

During the next 2424 hours, challenges will be released at regular intervals. The first two challenges will show up at 55 minutes into the contest. Then we will release one problem at each round o'clock, starting at the 11st hour and up to the 2323rd into the contest. Please keep an eye on the announcements page for updates.

All challenges have endured a very rigorous development cycle. We did our best to assure their quality. If you have a question regarding the challenges, or you believe that there is an error, please do not hesitate to contact the technical team who will be online at all times. You may contact the technical team by submitting a clarification request via the CSA Q&A system. Please keep in mind that we will do our best to answer questions, but we cannot comment on your code under any circumstances. Our first goal is to keep the integrity and fairness of the competition. We encourage you to attempt every problem, even if your solution is not optimal. You may be able to obtain a partial score by solving a subset of test files of a challenge. Please refer to the "Constraints and notes" section of each challenge for partial scoring data distribution.

Finally, a special thank you to all the volunteers who have made this IEEEXtreme possible, and we would like to express our sincerest appreciation to our sponsors and partners for believing in our ideas and supporting this great event.

Good luck and have fun!

Your first task is to write a program that outputs a message you'd like to say to the IEEEXtreme team, and submit it to the system.

Standard input

Nothing on standard input.

Standard output

Output a text with your message to us. The message should contain no more than 200200 words.

Constraints and notes

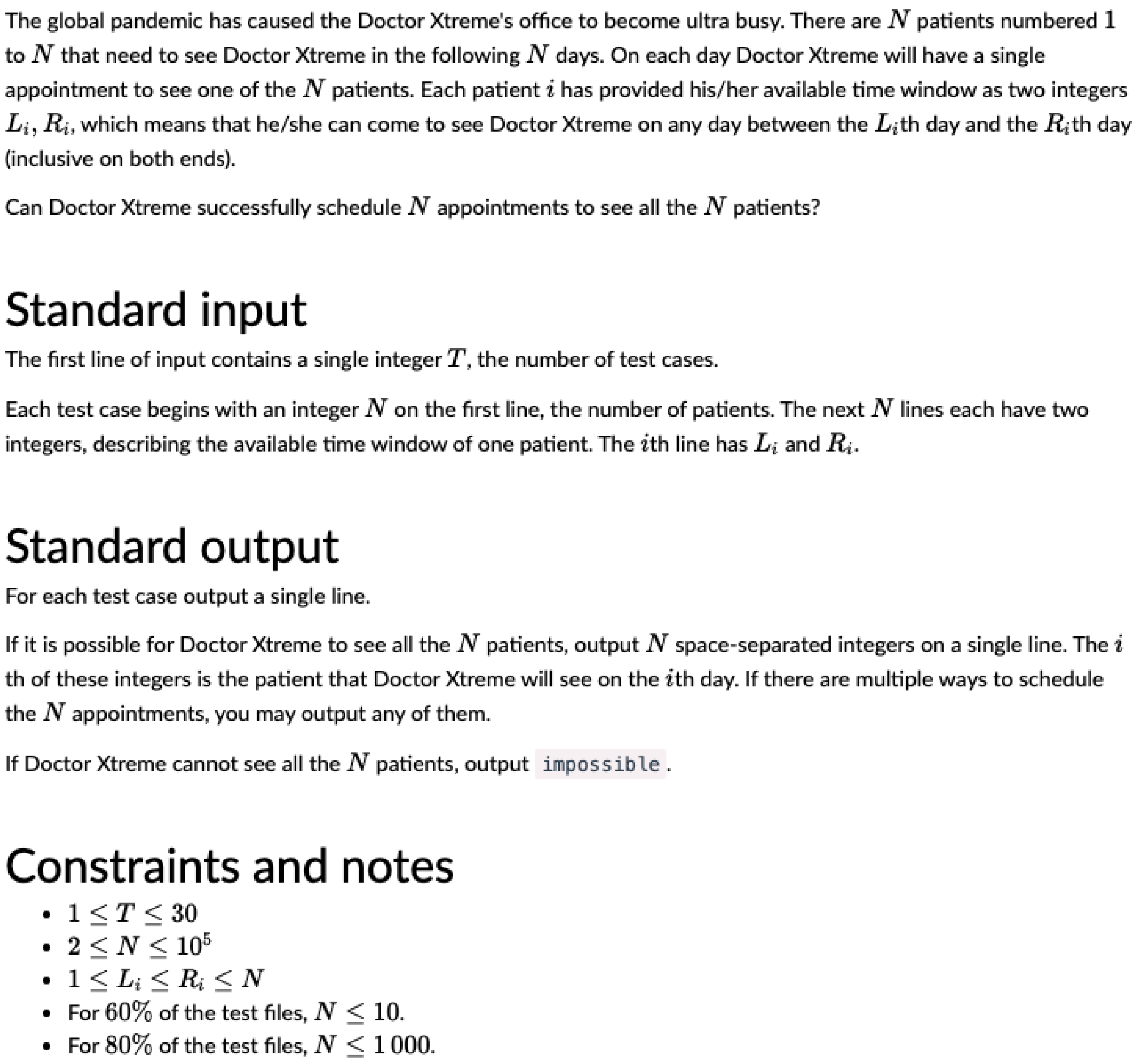
- This task will **not** be scored and it is **optional**.
- This "Constraints and notes" section is very important for each challenge. Please make sure that you read them.
- Please do take the time to write a message. We look forward to hearing from you!

Technical notes on Python

In this competition, if you are working with Python2 or Python3, it is recommended that you submit your solution using Pypy2 or Pypy3. Pypy runs much faster than the regular Python interpreter and may sometimes help avoid Time Limit Exceeded for an algorithm that has an expected time complexity.

Doctor's Appointments

Time limit: *7280 ms*
Memory limit: *264 MB*



The global pandemic has caused the Doctor Xtreme's office to become ultra busy. There are N patients numbered 1 to N that need to see Doctor Xtreme in the following N days. On each day Doctor Xtreme will have a single appointment to see one of the N patients. Each patient i has provided his/her available time window as two integers L_i, R_i, which means that he/she can come to see Doctor Xtreme on any day between the L_ith day and the R_ith day (inclusive on both ends).

Can Doctor Xtreme successfully schedule N appointments to see all the N patients?

Standard input

The first line of input contains a single integer T, the number of test cases.

Each test case begins with an integer N on the first line, the number of patients. The next N lines each have two integers, describing the available time window of one patient. The ith line has L_i and R_i.

Standard output

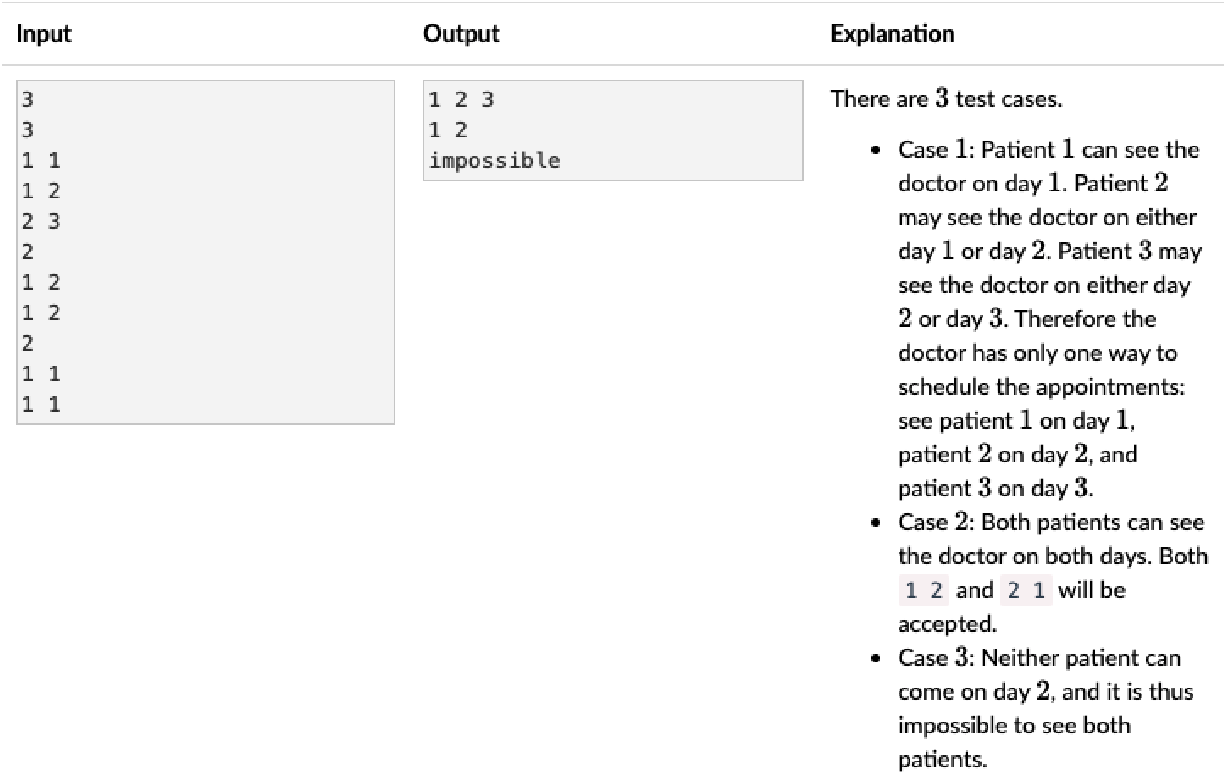
For each test case output a single line.

If it is possible for Doctor Xtreme to see all the N patients, output N space-separated integers on a single line. The i th of these integers is the patient that Doctor Xtreme will see on the ith day. If there are multiple ways to schedule the N appointments, you may output any of them.

If Doctor Xtreme cannot see all the N patients, output `impossible`.

Constraints and notes

- $1 \leq T \leq 30$
- $2 \leq N \leq 10^5$
- $1 \leq L_i \leq R_i \leq N$
- For 60% of the test files, $N \leq 10$.
- For 80% of the test files, $N \leq 1\,000$.

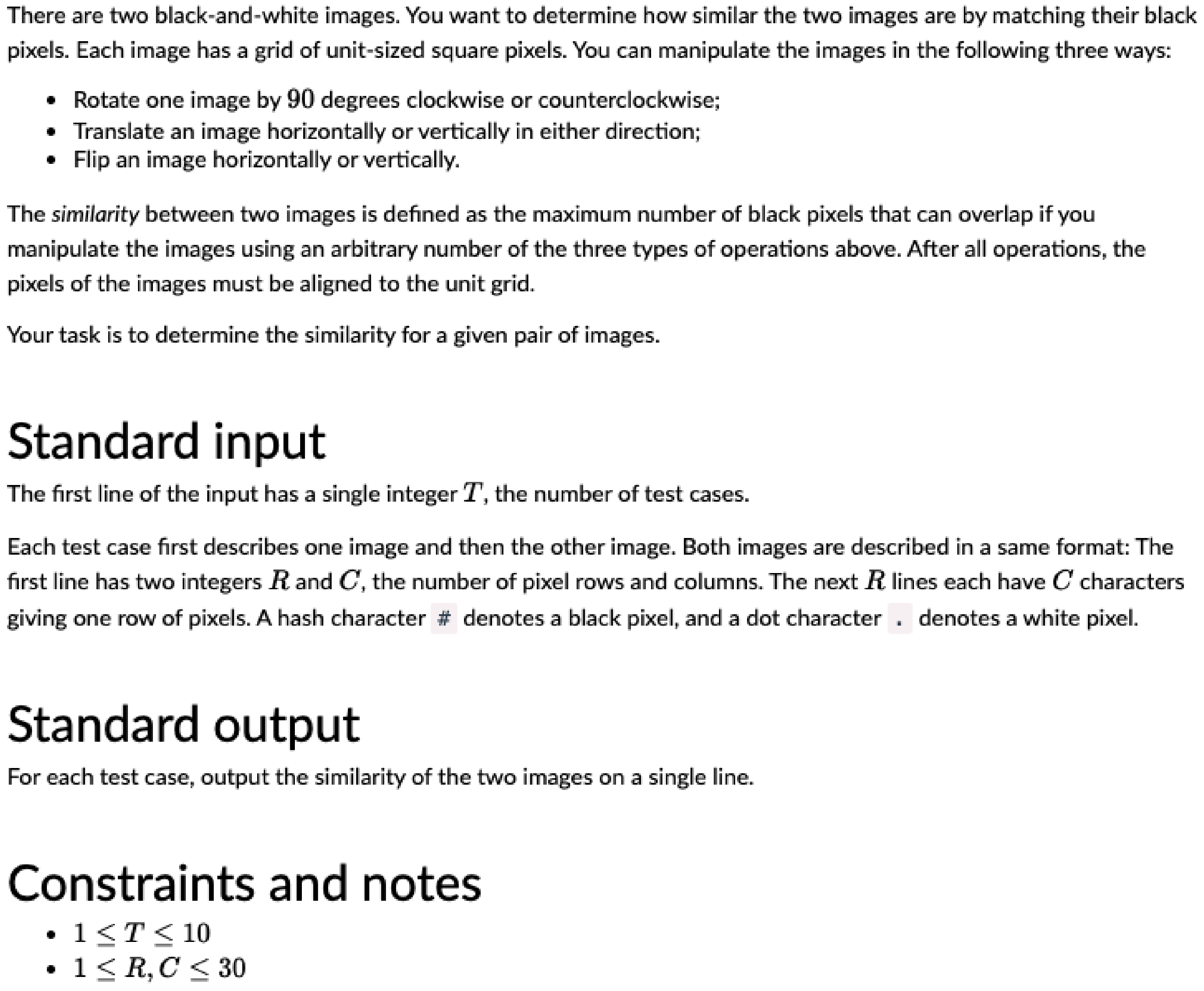


Input	**Output**	**Explanation**

3
3
1 1
1 2
2 3
2
1 2
1 2
2
1 1
1 1
	1 2 3
	1 2
	impossible
		There are 3 test cases.
		- Case 1: Patient 1 can see the doctor on day 1. Patient 2 may see the doctor on either day 1 or day 2. Patient 3 may see the doctor on either day 2 or day 3. Therefore the doctor has only one way to schedule the appointments: see patient 1 on day 1, patient 2 on day 2, and patient 3 on day 3.
		- Case 2: Both patients can see the doctor on both days. Both `1 2` and `2 1` will be accepted.
		- Case 3: Neither patient can come on day 2, and it is thus impossible to see both patients.

Image Similarity

Time limit: *4880 ms*
Memory limit: *264 MB*



There are two black-and-white images. You want to determine how similar the two images are by matching their black pixels. Each image has a grid of unit-sized square pixels. You can manipulate the images in the following three ways:

- Rotate one image by 90 degrees clockwise or counterclockwise;
- Translate an image horizontally or vertically in either direction;
- Flip an image horizontally or vertically.

The *similarity* between two images is defined as the maximum number of black pixels that can overlap if you manipulate the images using an arbitrary number of the three types of operations above. After all operations, the pixels of the images must be aligned to the unit grid.

Your task is to determine the similarity for a given pair of images.

Standard input

The first line of the input has a single integer T, the number of test cases.

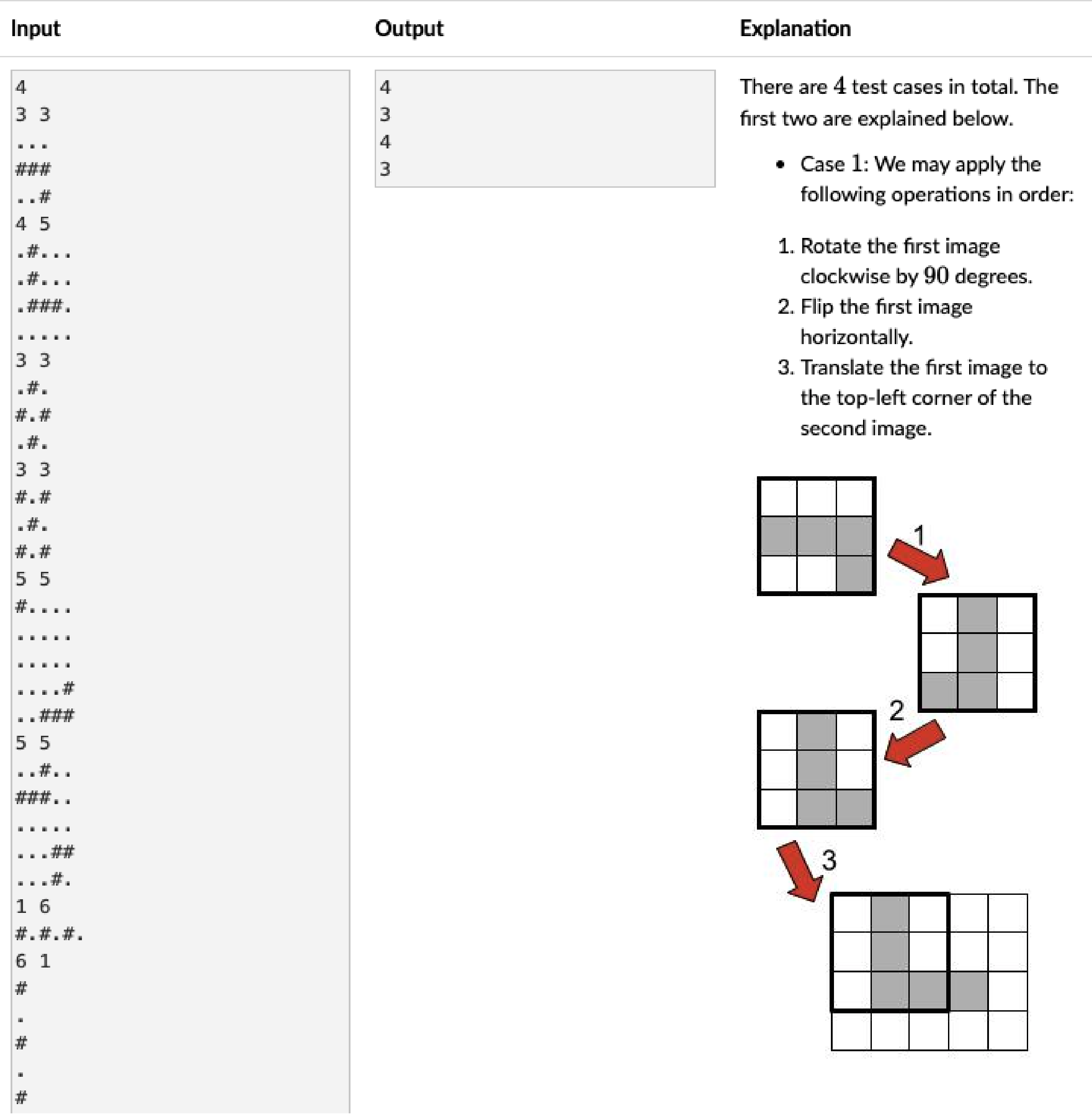
Each test case first describes one image and then the other image. Both images are described in a same format: The first line has two integers R and C, the number of pixel rows and columns. The next R lines each have C characters giving one row of pixels. A hash character `#` denotes a black pixel, and a dot character `.` denotes a white pixel.

Standard output

For each test case, output the similarity of the two images on a single line.

Constraints and notes

- $1 \leq T \leq 10$
- $1 \leq R, C \leq 30$



Input

```
4
3 3
...
###
..#
4 5
.#...
.#...
.###.
.....
3 3
.#.
#.#
.#.
3 3
#.#
.#.
#.#
5 5
#....
.....
.....
....#
..###
5 5
..#..
###..
.....
...##
...#.
1 6
#.#.#.
6 1
#
.
#
.
#
```

Output

```
4
3
4
3
```

Explanation

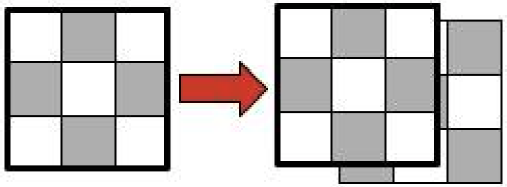
There are 4 test cases in total. The first two are explained below.

- Case 1: We may apply the following operations in order:

1. Rotate the first image clockwise by 90 degrees.
2. Flip the first image horizontally.
3. Translate the first image to the top-left corner of the second image.

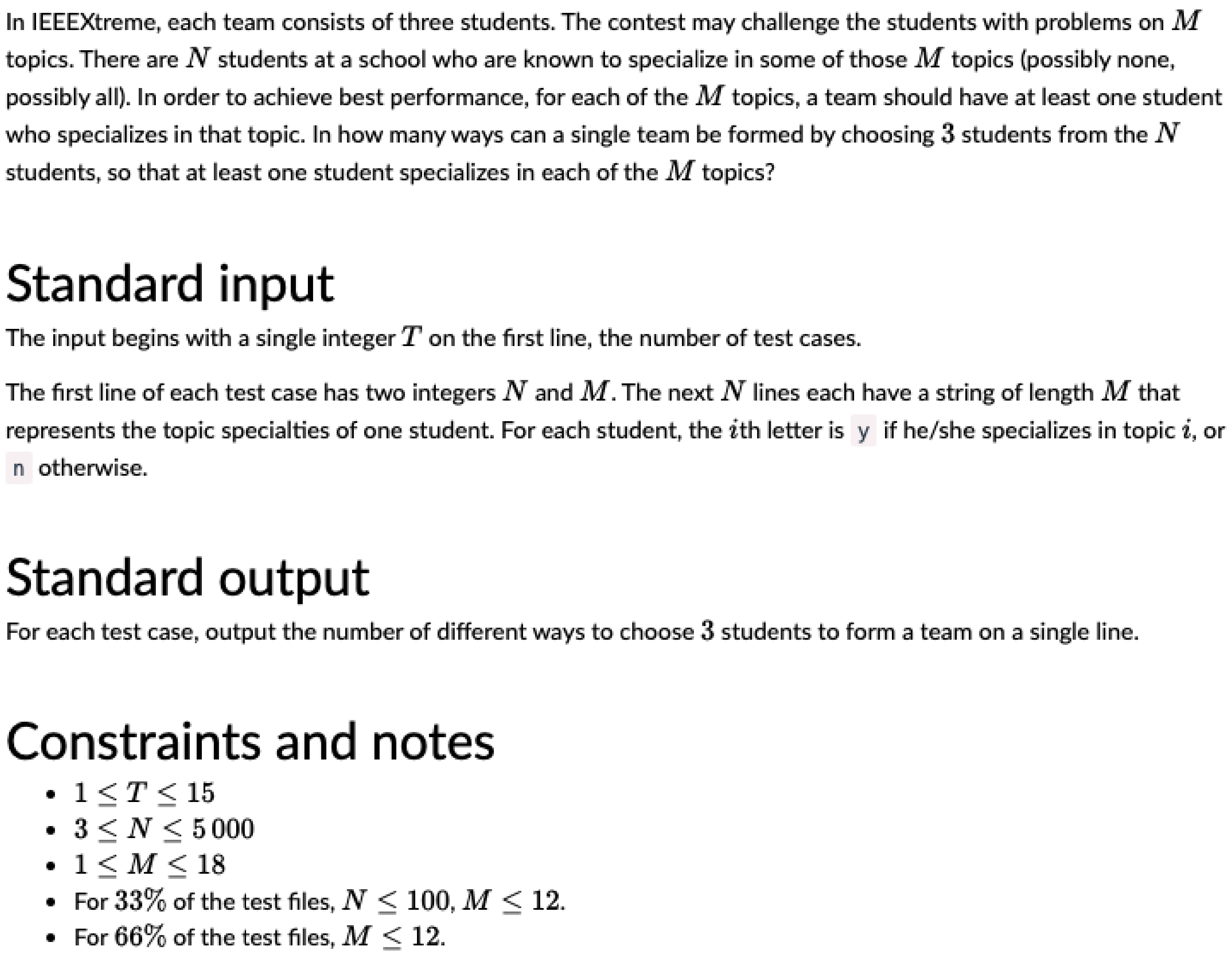
```
.
#
.
```

- Case 2: we can translate the first image as follows. Note that one image may extend beyond the boundary of the other image when we match the black pixels.



Xtreme Teams

Time limit: *2480 ms*
Memory limit: *264 MB*



In IEEEXtreme, each team consists of three students. The contest may challenge the students with problems on M topics. There are N students at a school who are known to specialize in some of those M topics (possibly none, possibly all). In order to achieve best performance, for each of the M topics, a team should have at least one student who specializes in that topic. In how many ways can a single team be formed by choosing 3 students from the N students, so that at least one student specializes in each of the M topics?

Standard input

The input begins with a single integer T on the first line, the number of test cases.

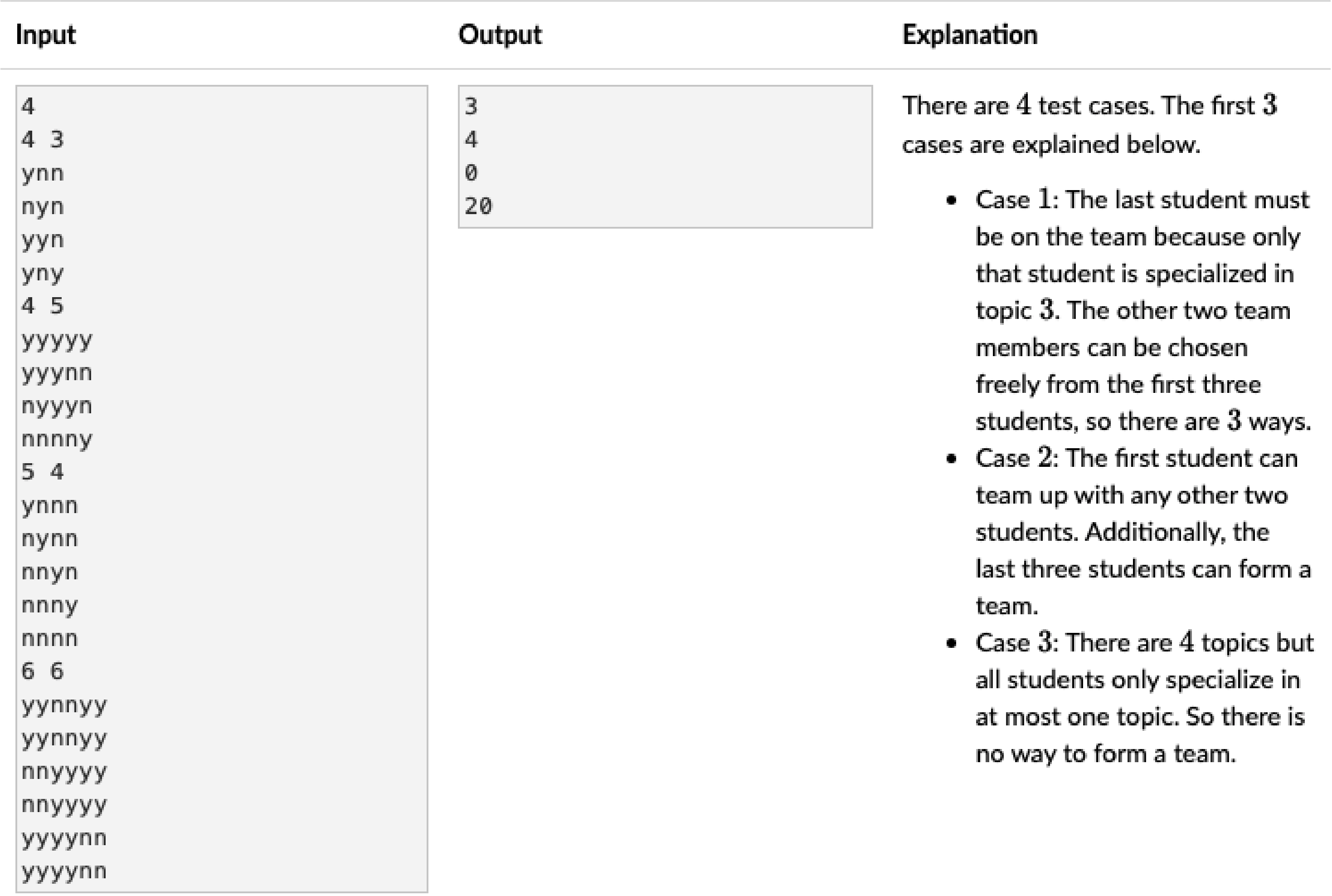
The first line of each test case has two integers N and M. The next N lines each have a string of length M that represents the topic specialties of one student. For each student, the ith letter is `y` if he/she specializes in topic i, or `n` otherwise.

Standard output

For each test case, output the number of different ways to choose 3 students to form a team on a single line.

Constraints and notes

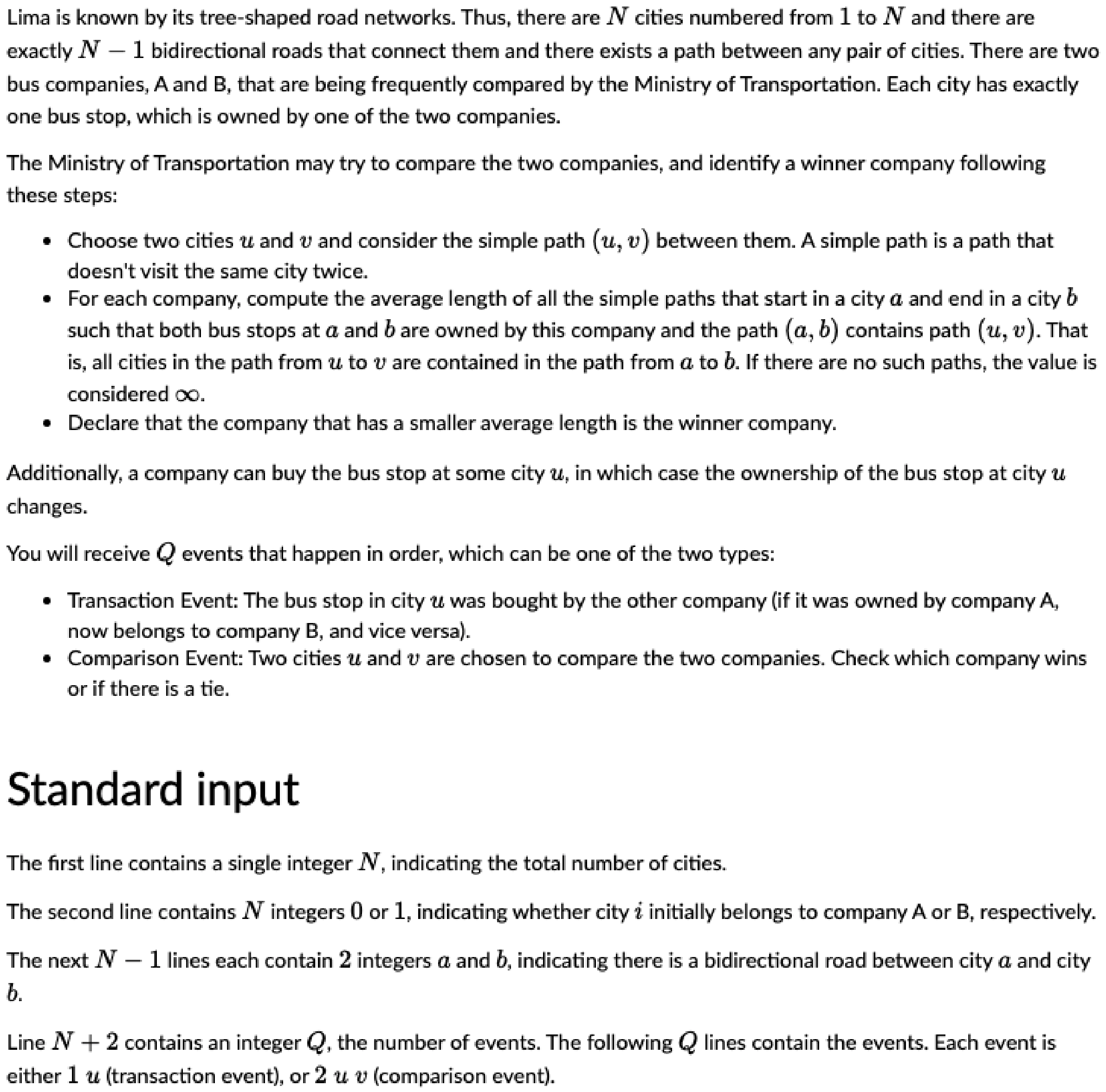
- $1 \leq T \leq 15$
- $3 \leq N \leq 5\,000$
- $1 \leq M \leq 18$
- For 33% of the test files, $N \leq 100$, $M \leq 12$.
- For 66% of the test files, $M \leq 12$.



Input	Output	Explanation
4 4 3 ynn nyn yyn yny 4 5 yyyyy yyynn nyyyn nnnny 5 4 ynnn nynn nnyn nnny nnnn 6 6 yynnyy yynnyy nnyyyy nnyyyy yyyynn yyyynn	3 4 0 20	There are 4 test cases. The first 3 cases are explained below. • Case 1: The last student must be on the team because only that student is specialized in topic 3. The other two team members can be chosen freely from the first three students, so there are 3 ways. • Case 2: The first student can team up with any other two students. Additionally, the last three students can form a team. • Case 3: There are 4 topics but all students only specialize in at most one topic. So there is no way to form a team.

Bus Company

Time limit: *7280 ms*
Memory limit: *264 MB*



Lima is known by its tree-shaped road networks. Thus, there are N cities numbered from 1 to N and there are exactly $N-1$ bidirectional roads that connect them and there exists a path between any pair of cities. There are two bus companies, A and B, that are being frequently compared by the Ministry of Transportation. Each city has exactly one bus stop, which is owned by one of the two companies.

The Ministry of Transportation may try to compare the two companies, and identify a winner company following these steps:

- Choose two cities u and v and consider the simple path (u, v) between them. A simple path is a path that doesn't visit the same city twice.
- For each company, compute the average length of all the simple paths that start in a city a and end in a city b such that both bus stops at a and b are owned by this company and the path (a, b) contains path (u, v). That is, all cities in the path from u to v are contained in the path from a to b. If there are no such paths, the value is considered ∞.
- Declare that the company that has a smaller average length is the winner company.

Additionally, a company can buy the bus stop at some city u, in which case the ownership of the bus stop at city u changes.

You will receive Q events that happen in order, which can be one of the two types:

- Transaction Event: The bus stop in city u was bought by the other company (if it was owned by company A, now belongs to company B, and vice versa).
- Comparison Event: Two cities u and v are chosen to compare the two companies. Check which company wins or if there is a tie.

Standard input

The first line contains a single integer N, indicating the total number of cities.

The second line contains N integers 0 or 1, indicating whether city i initially belongs to company A or B, respectively.

The next $N-1$ lines each contain 2 integers a and b, indicating there is a bidirectional road between city a and city b.

Line $N+2$ contains an integer Q, the number of events. The following Q lines contain the events. Each event is either 1 u (transaction event), or 2 u v (comparison event).

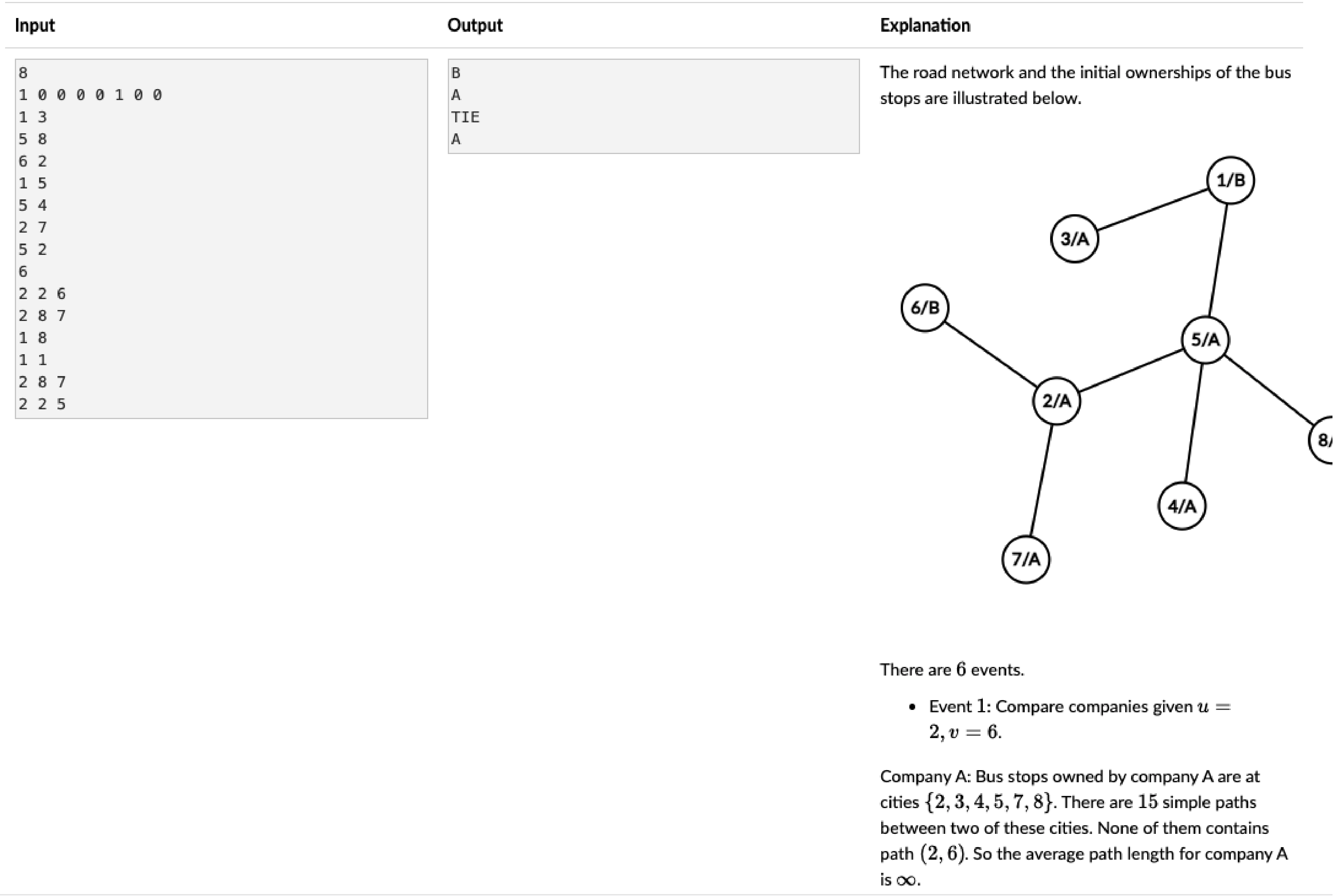
Standard Output

For each comparison event print `A` if company A wins, `B` if company B wins, or `TIE` if there is a tie between both companies.

Constraints and notes

- $N \geq 2$
- $N, Q \leq 2 \cdot 10^5$
- In all events, $1 \leq u, v \leq N, u \neq v$.
- The road network forms a tree.
- There is at least one comparison event.

- For 5% of the test files, $N, Q \leq 10^3$.
- For 10% of the test files, $N, Q \leq 10^4$.
- For 30% of the test files, $N, Q \leq 10^5$.



Input

```
8
1 0 0 0 0 1 0 0
1 3
5 8
6 2
1 5
5 4
2 7
5 2
6
2 2 6
2 8 7
1 8
1 1
2 8 7
2 2 5
```

Output

```
B
A
TIE
A
```

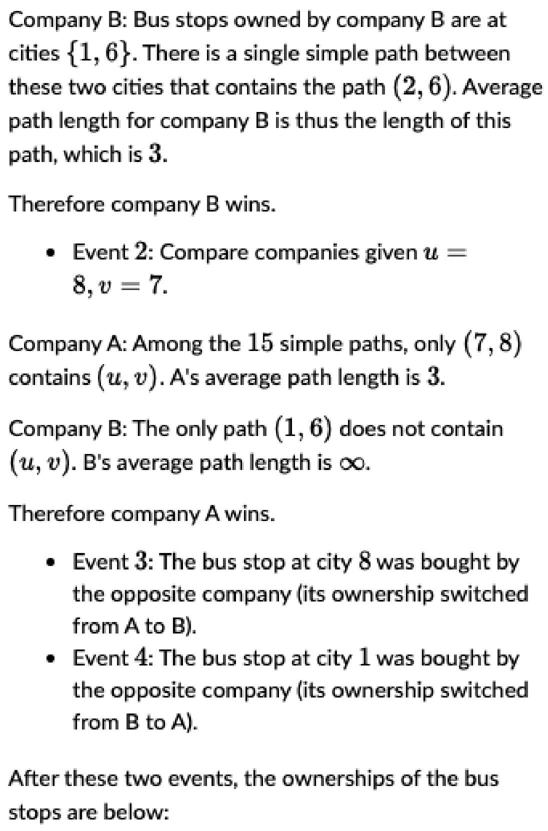
Explanation

The road network and the initial ownerships of the bus stops are illustrated below.

There are 6 events.

- Event 1: Compare companies given $u = 2, v = 6$.

Company A: Bus stops owned by company A are at cities $\{2, 3, 4, 5, 7, 8\}$. There are 15 simple paths between two of these cities. None of them contains path $(2, 6)$. So the average path length for company A is ∞.



Company B: Bus stops owned by company B are at cities $\{1, 6\}$. There is a single simple path between these two cities that contains the path $(2, 6)$. Average path length for company B is thus the length of this path, which is 3.

Therefore company B wins.

- Event 2: Compare companies given $u = 8, v = 7$.

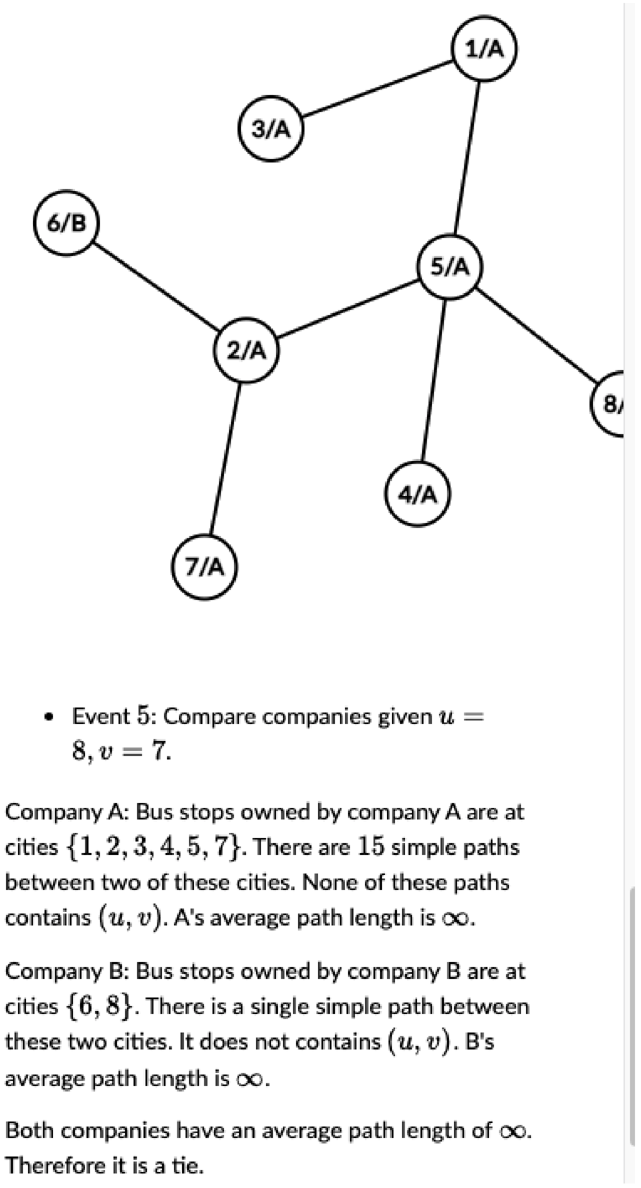
Company A: Among the 15 simple paths, only $(7, 8)$ contains (u, v). A's average path length is 3.

Company B: The only path $(1, 6)$ does not contain (u, v). B's average path length is ∞.

Therefore company A wins.

- Event 3: The bus stop at city 8 was bought by the opposite company (its ownership switched from A to B).
- Event 4: The bus stop at city 1 was bought by the opposite company (its ownership switched from B to A).

After these two events, the ownerships of the bus stops are below:

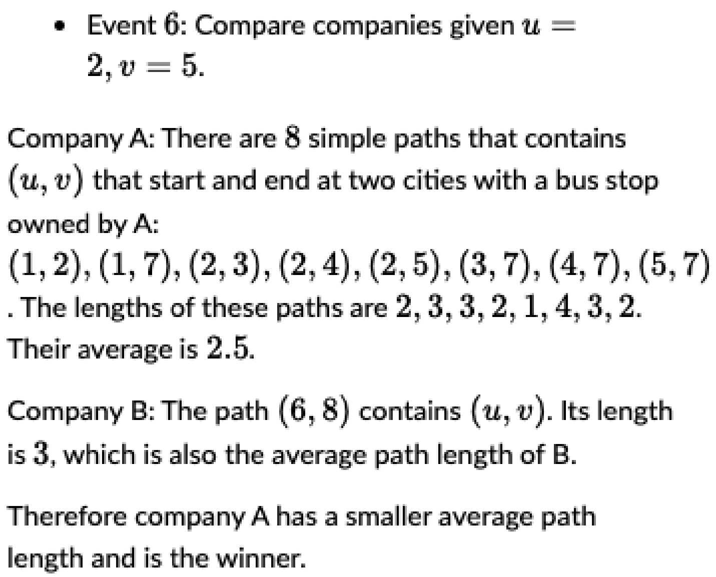


- Event 5: Compare companies given $u = 8, v = 7$.

Company A: Bus stops owned by company A are at cities $\{1, 2, 3, 4, 5, 7\}$. There are 15 simple paths between two of these cities. None of these paths contains (u, v). A's average path length is ∞.

Company B: Bus stops owned by company B are at cities $\{6, 8\}$. There is a single simple path between these two cities. It does not contains (u, v). B's average path length is ∞.

Both companies have an average path length of ∞. Therefore it is a tie.



- Event 6: Compare companies given $u = 2, v = 5$.

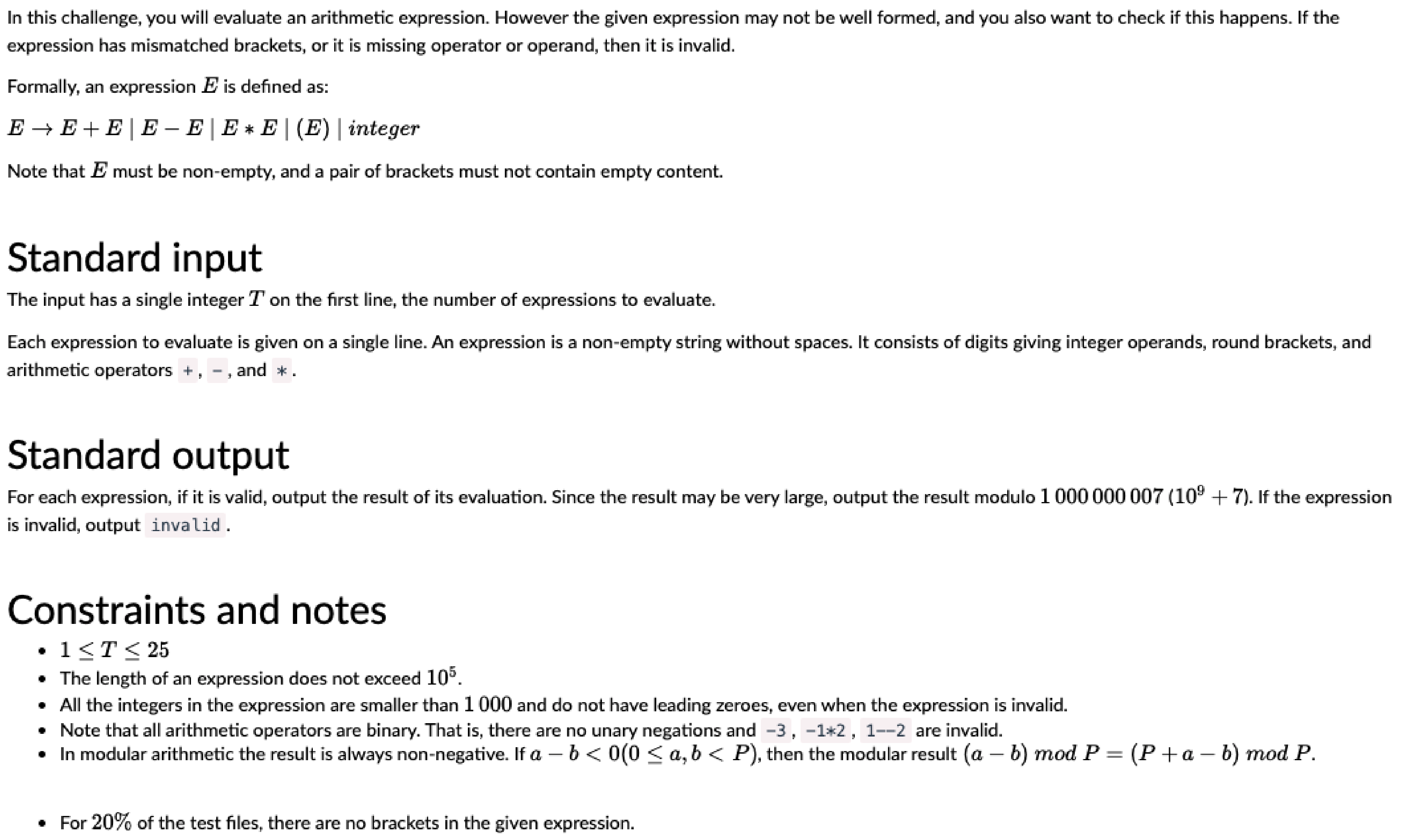
Company A: There are 8 simple paths that contains (u, v) that start and end at two cities with a bus stop owned by A: $(1,2), (1,7), (2,3), (2,4), (2,5), (3,7), (4,7), (5,7)$. The lengths of these paths are $2, 3, 3, 2, 1, 4, 3, 2$. Their average is 2.5.

Company B: The path $(6, 8)$ contains (u, v). Its length is 3, which is also the average path length of B.

Therefore company A has a smaller average path length and is the winner.

Expression Evaluation

Time limit: *4880 ms*
Memory limit: *264 MB*



In this challenge, you will evaluate an arithmetic expression. However the given expression may not be well formed, and you also want to check if this happens. If the expression has mismatched brackets, or it is missing operator or operand, then it is invalid.

Formally, an expression E is defined as:

$$E \rightarrow E + E \mid E - E \mid E * E \mid (E) \mid integer$$

Note that E must be non-empty, and a pair of brackets must not contain empty content.

Standard input

The input has a single integer T on the first line, the number of expressions to evaluate.

Each expression to evaluate is given on a single line. An expression is a non-empty string without spaces. It consists of digits giving integer operands, round brackets, and arithmetic operators `+`, `-`, and `*`.

Standard output

For each expression, if it is valid, output the result of its evaluation. Since the result may be very large, output the result modulo $1\,000\,000\,007$ $(10^9 + 7)$. If the expression is invalid, output `invalid`.

Constraints and notes

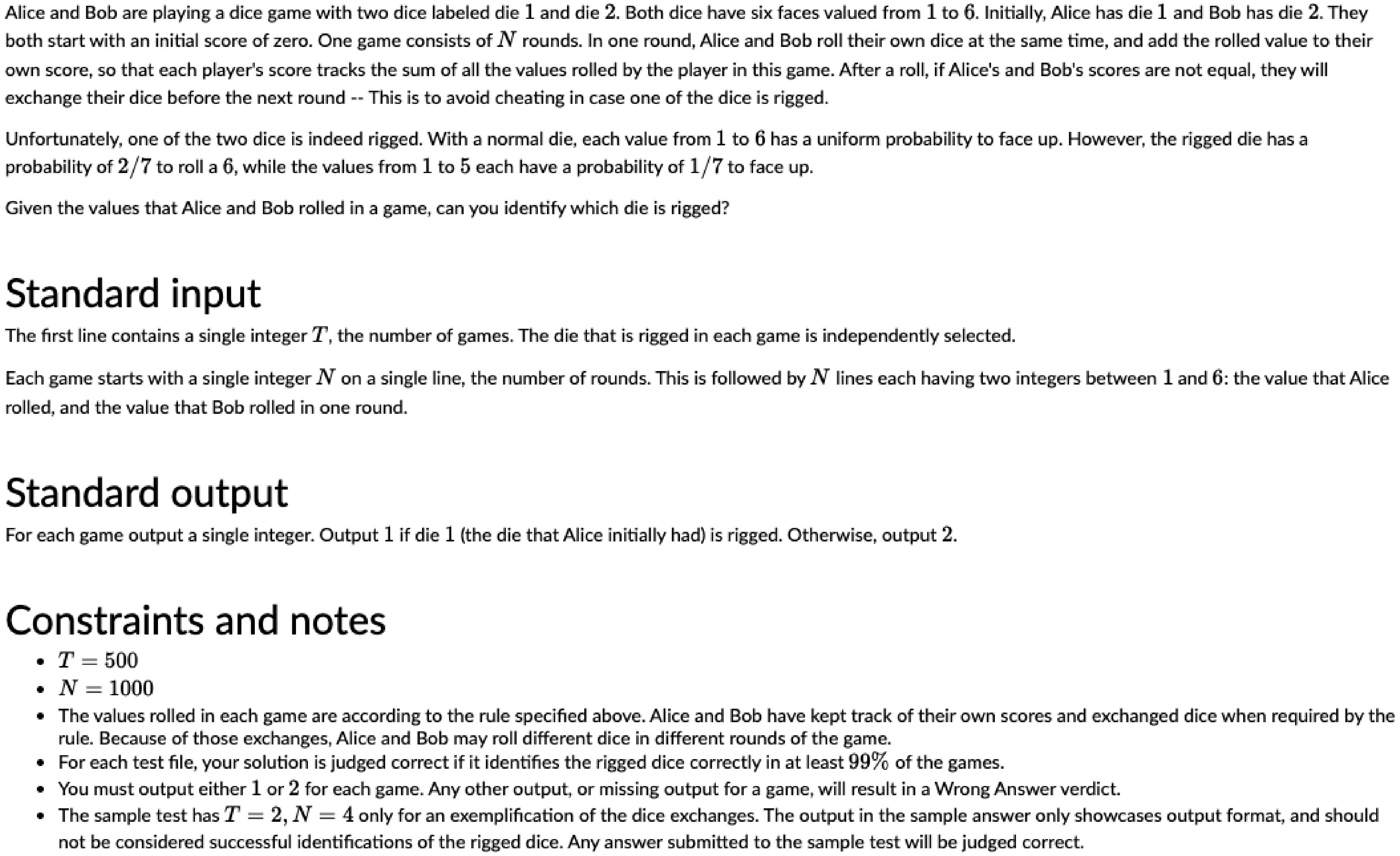
- $1 \leq T \leq 25$
- The length of an expression does not exceed 10^5.
- All the integers in the expression are smaller than $1\,000$ and do not have leading zeroes, even when the expression is invalid.
- Note that all arithmetic operators are binary. That is, there are no unary negations and `-3`, `-1*2`, `1--2` are invalid.
- In modular arithmetic the result is always non-negative. If $a - b < 0 (0 \leq a, b < P)$, then the modular result $(a - b)\ mod\ P = (P + a - b)\ mod\ P$.

- For 20% of the test files, there are no brackets in the given expression.

Input	Output	Explanation
4 1+2*3+4 1*2-3 123 +-*	11 1000000006 123 invalid	There are 4 test cases without round brackets. • Case 1: It is straightforward to evaluate that $1+2\times 3+4=11$. • Case 2: $1\times 2-3=-1$. In modular arithmetic, -1 is $1\,000\,000\,006$ (mod $1\,000\,000\,007$). • Case 3: By definition $E \to integer$. Therefore 123 evaluates to 123. • Case 4: Each operator must have operands. This expression is invalid.
4 (1+2)*3 () ()100 (((1)))	9 invalid invalid 1	There are 4 test cases with round brackets. • Case 1: It is straightforward to evaluate that $(1+2)\times 3=9$. • Case 2: Empty brackets are not allowed by definition. • Case 3: Not only empty brackets are disallowed, but also there misses an operator between $()$ and 100. • Case 4: By definition $E \to (E)$. Therefore $E \to (E) \to ((E)) \to (((E))) \to (((1))))$, which evaluates to 1.

Rigged Dice

Time limit: *2480 ms*
Memory limit: *264 MB*



Alice and Bob are playing a dice game with two dice labeled die 1 and die 2. Both dice have six faces valued from 1 to 6. Initially, Alice has die 1 and Bob has die 2. They both start with an initial score of zero. One game consists of N rounds. In one round, Alice and Bob roll their own dice at the same time, and add the rolled value to their own score, so that each player's score tracks the sum of all the values rolled by the player in this game. After a roll, if Alice's and Bob's scores are not equal, they will exchange their dice before the next round -- This is to avoid cheating in case one of the dice is rigged.

Unfortunately, one of the two dice is indeed rigged. With a normal die, each value from 1 to 6 has a uniform probability to face up. However, the rigged die has a probability of $2/7$ to roll a 6, while the values from 1 to 5 each have a probability of $1/7$ to face up.

Given the values that Alice and Bob rolled in a game, can you identify which die is rigged?

Standard input

The first line contains a single integer T, the number of games. The die that is rigged in each game is independently selected.

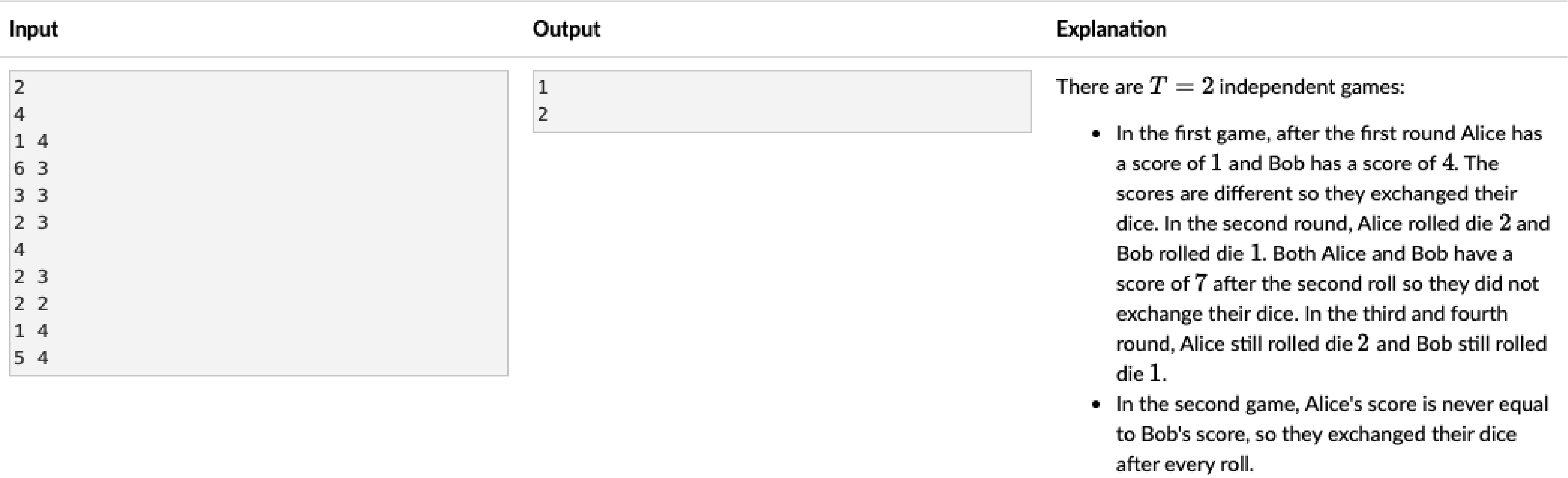
Each game starts with a single integer N on a single line, the number of rounds. This is followed by N lines each having two integers between 1 and 6: the value that Alice rolled, and the value that Bob rolled in one round.

Standard output

For each game output a single integer. Output 1 if die 1 (the die that Alice initially had) is rigged. Otherwise, output 2.

Constraints and notes

- $T = 500$
- $N = 1000$
- The values rolled in each game are according to the rule specified above. Alice and Bob have kept track of their own scores and exchanged dice when required by the rule. Because of those exchanges, Alice and Bob may roll different dice in different rounds of the game.
- For each test file, your solution is judged correct if it identifies the rigged dice correctly in at least 99% of the games.
- You must output either 1 or 2 for each game. Any other output, or missing output for a game, will result in a Wrong Answer verdict.
- The sample test has $T = 2, N = 4$ only for an exemplification of the dice exchanges. The output in the sample answer only showcases output format, and should not be considered successful identifications of the rigged dice. Any answer submitted to the sample test will be judged correct.

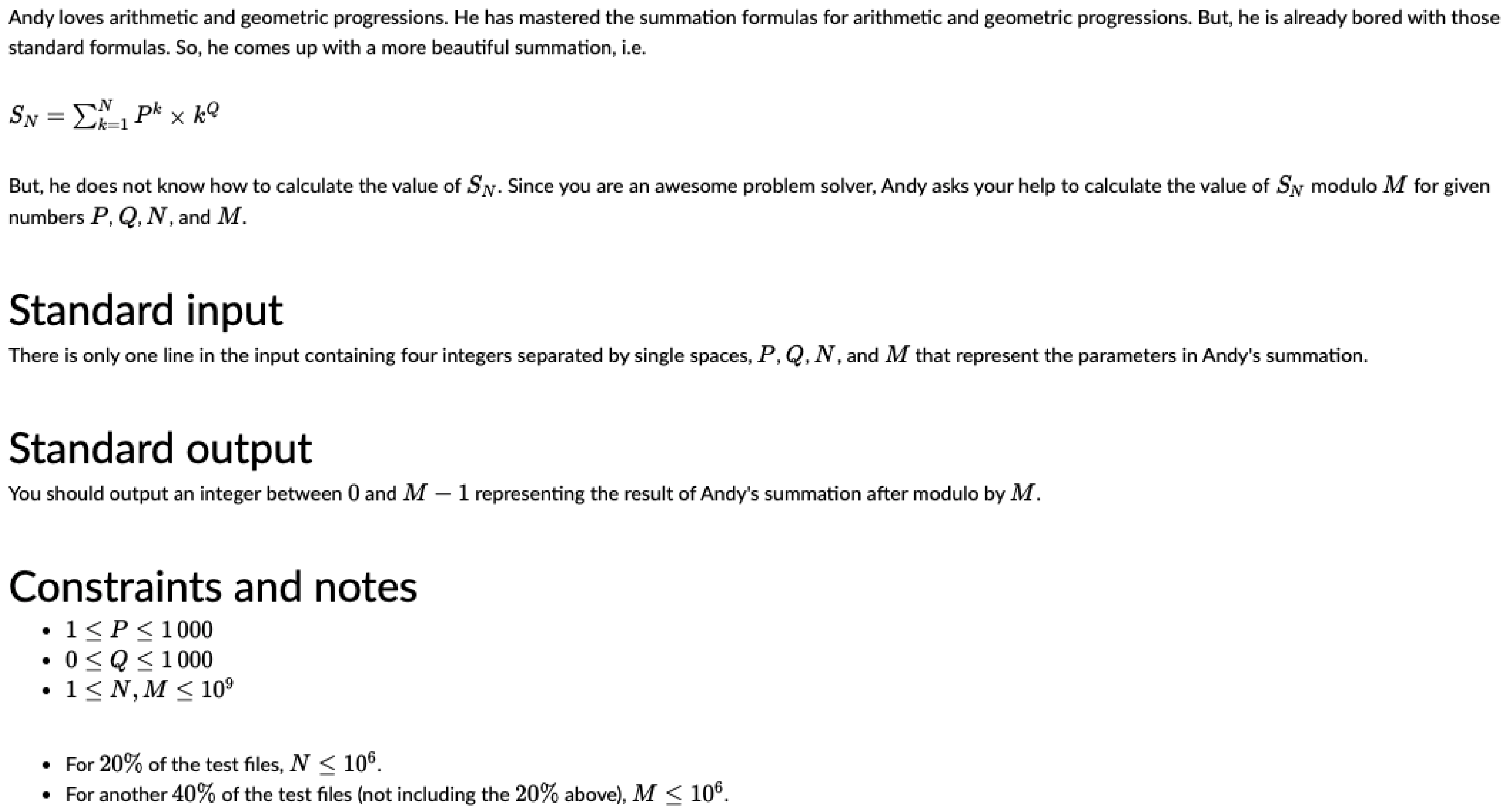


Input	Output	Explanation

2
4
1 4
6 3
3 3
2 3
4
2 3
2 2
1 4
5 4
	1
	2
		There are $T = 2$ independent games:
		- In the first game, after the first round Alice has a score of 1 and Bob has a score of 4. The scores are different so they exchanged their dice. In the second round, Alice rolled die 2 and Bob rolled die 1. Both Alice and Bob have a score of 7 after the second roll so they did not exchange their dice. In the third and fourth round, Alice still rolled die 2 and Bob still rolled die 1.
		- In the second game, Alice's score is never equal to Bob's score, so they exchanged their dice after every roll.

Beautiful Summation

Time limit: *2480 ms*
Memory limit: *264 MB*



Andy loves arithmetic and geometric progressions. He has mastered the summation formulas for arithmetic and geometric progressions. But, he is already bored with those standard formulas. So, he comes up with a more beautiful summation, i.e.

$S_N = \sum_{k=1}^{N} P^k \times k^Q$

But, he does not know how to calculate the value of S_N. Since you are an awesome problem solver, Andy asks your help to calculate the value of S_N modulo M for given numbers P, Q, N, and M.

Standard input

There is only one line in the input containing four integers separated by single spaces, P, Q, N, and M that represent the parameters in Andy's summation.

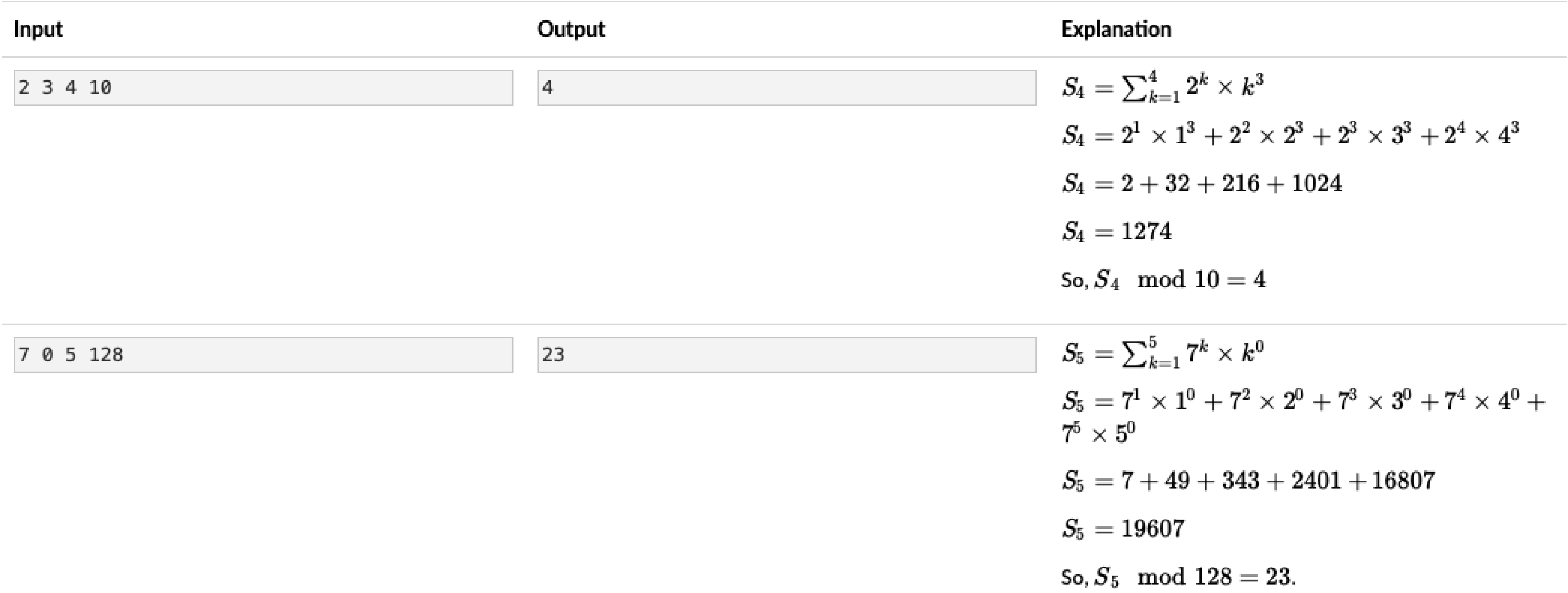
Standard output

You should output an integer between 0 and $M - 1$ representing the result of Andy's summation after modulo by M.

Constraints and notes

- $1 \leq P \leq 1\,000$
- $0 \leq Q \leq 1\,000$
- $1 \leq N, M \leq 10^9$

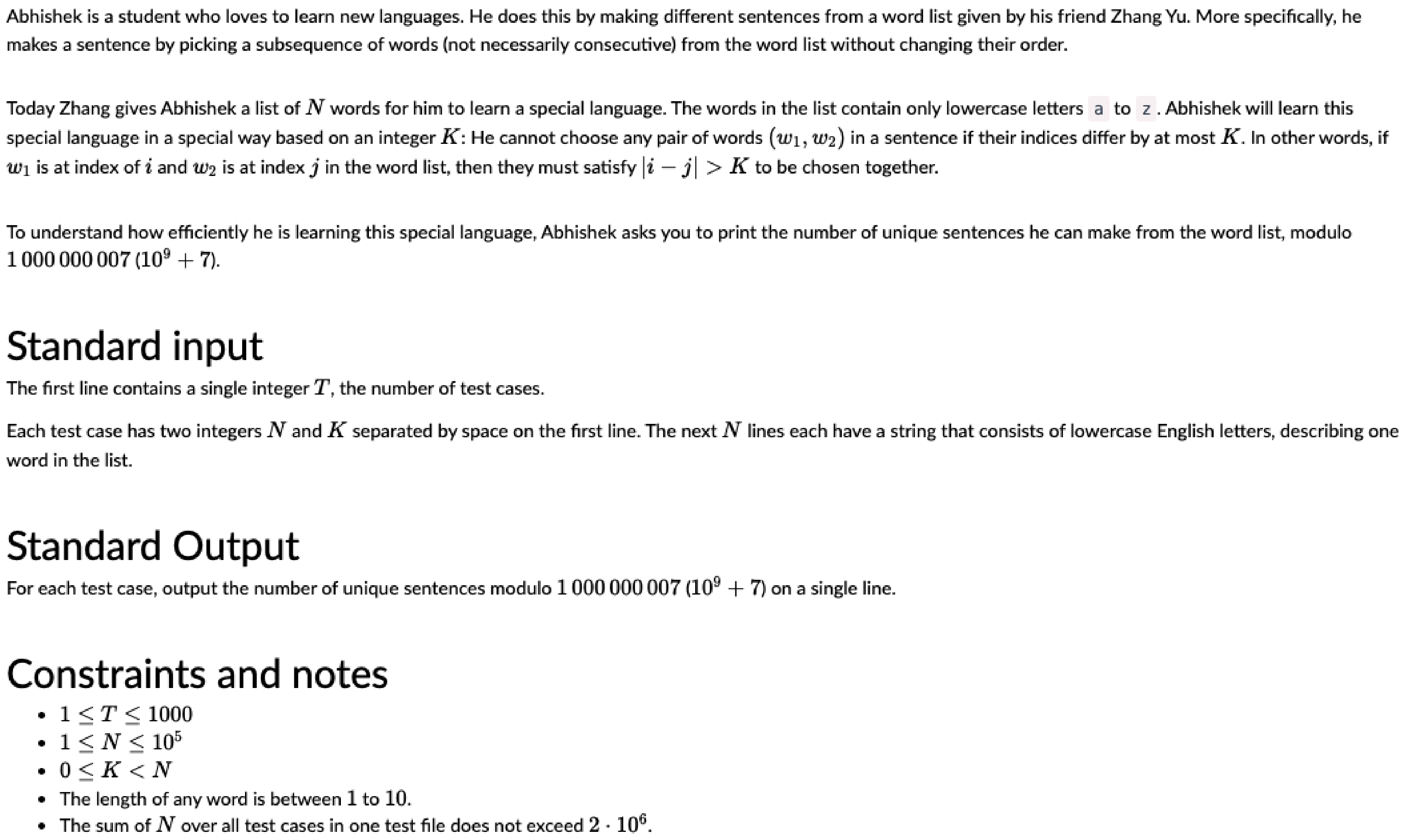
- For 20% of the test files, $N \leq 10^6$.
- For another 40% of the test files (not including the 20% above), $M \leq 10^6$.



Input	Output	Explanation
`2 3 4 10`	`4`	$S_4 = \sum_{k=1}^{4} 2^k \times k^3$ $S_4 = 2^1 \times 1^3 + 2^2 \times 2^3 + 2^3 \times 3^3 + 2^4 \times 4^3$ $S_4 = 2 + 32 + 216 + 1024$ $S_4 = 1274$ So, $S_4 \mod 10 = 4$
`7 0 5 128`	`23`	$S_5 = \sum_{k=1}^{5} 7^k \times k^0$ $S_5 = 7^1 \times 1^0 + 7^2 \times 2^0 + 7^3 \times 3^0 + 7^4 \times 4^0 + 7^5 \times 5^0$ $S_5 = 7 + 49 + 343 + 2401 + 16807$ $S_5 = 19607$ So, $S_5 \mod 128 = 23$.

Language Learning

Time limit: *2480 ms*
Memory limit: *264 MB*



Abhishek is a student who loves to learn new languages. He does this by making different sentences from a word list given by his friend Zhang Yu. More specifically, he makes a sentence by picking a subsequence of words (not necessarily consecutive) from the word list without changing their order.

Today Zhang gives Abhishek a list of N words for him to learn a special language. The words in the list contain only lowercase letters `a` to `z`. Abhishek will learn this special language in a special way based on an integer K: He cannot choose any pair of words (w_1, w_2) in a sentence if their indices differ by at most K. In other words, if w_1 is at index of i and w_2 is at index j in the word list, then they must satisfy $|i - j| > K$ to be chosen together.

To understand how efficiently he is learning this special language, Abhishek asks you to print the number of unique sentences he can make from the word list, modulo $1\,000\,000\,007$ ($10^9 + 7$).

Standard input

The first line contains a single integer T, the number of test cases.

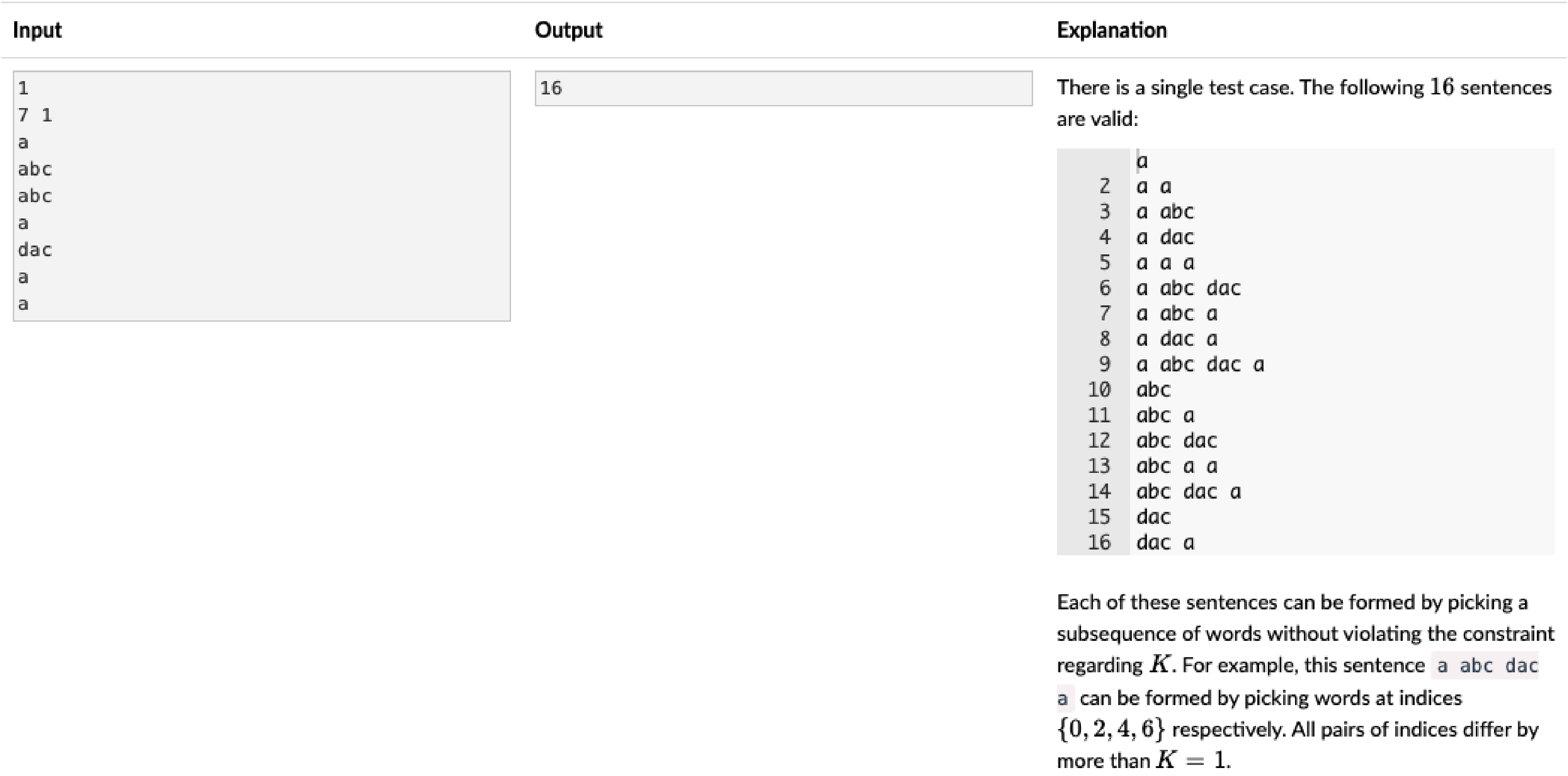
Each test case has two integers N and K separated by space on the first line. The next N lines each have a string that consists of lowercase English letters, describing one word in the list.

Standard Output

For each test case, output the number of unique sentences modulo $1\,000\,000\,007$ ($10^9 + 7$) on a single line.

Constraints and notes

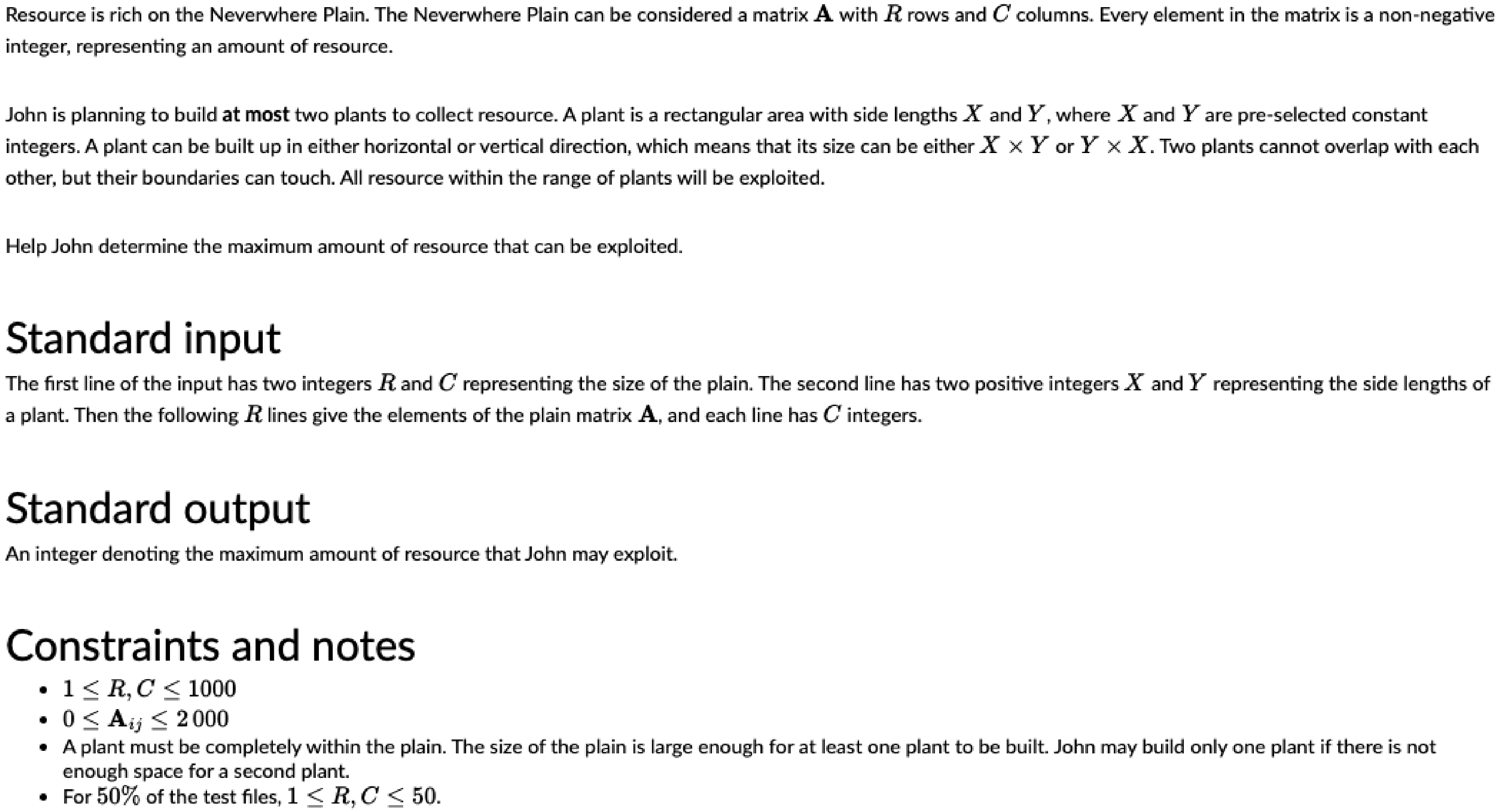
- $1 \leq T \leq 1000$
- $1 \leq N \leq 10^5$
- $0 \leq K < N$
- The length of any word is between 1 to 10.
- The sum of N over all test cases in one test file does not exceed $2 \cdot 10^6$.



Input	Output	Explanation
1 7 1 a abc abc a dac a a	16	There is a single test case. The following 16 sentences are valid: a a a a abc a dac a a a a abc dac a abc a a dac a a abc dac a abc abc a abc dac abc a a abc dac a dac dac a Each of these sentences can be formed by picking a subsequence of words without violating the constraint regarding K. For example, this sentence `a abc dac a` can be formed by picking words at indices $\{0, 2, 4, 6\}$ respectively. All pairs of indices differ by more than $K = 1$.

Maximum Exploitation

Time limit: *2480 ms*
Memory limit: *264 MB*



Resource is rich on the Neverwhere Plain. The Neverwhere Plain can be considered a matrix $\mathbf{A}$ with R rows and C columns. Every element in the matrix is a non-negative integer, representing an amount of resource.

John is planning to build **at most** two plants to collect resource. A plant is a rectangular area with side lengths X and Y, where X and Y are pre-selected constant integers. A plant can be built up in either horizontal or vertical direction, which means that its size can be either $X \times Y$ or $Y \times X$. Two plants cannot overlap with each other, but their boundaries can touch. All resource within the range of plants will be exploited.

Help John determine the maximum amount of resource that can be exploited.

Standard input

The first line of the input has two integers R and C representing the size of the plain. The second line has two positive integers X and Y representing the side lengths of a plant. Then the following R lines give the elements of the plain matrix $\mathbf{A}$, and each line has C integers.

Standard output

An integer denoting the maximum amount of resource that John may exploit.

Constraints and notes

- $1 \leq R, C \leq 1000$
- $0 \leq \mathbf{A}_{ij} \leq 2\,000$
- A plant must be completely within the plain. The size of the plain is large enough for at least one plant to be built. John may build only one plant if there is not enough space for a second plant.
- For 50% of the test files, $1 \leq R, C \leq 50$.

Input

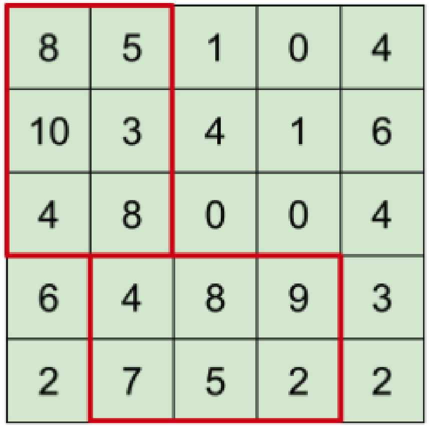
```
5 5
3 2
8 5 1 0 4
10 3 4 1 6
4 8 0 0 4
6 4 8 9 3
2 7 5 2 2
```

Output

```
73
```

Explanation

The plant area is 3×2 or 2×3. The two plants can be built at these locations:

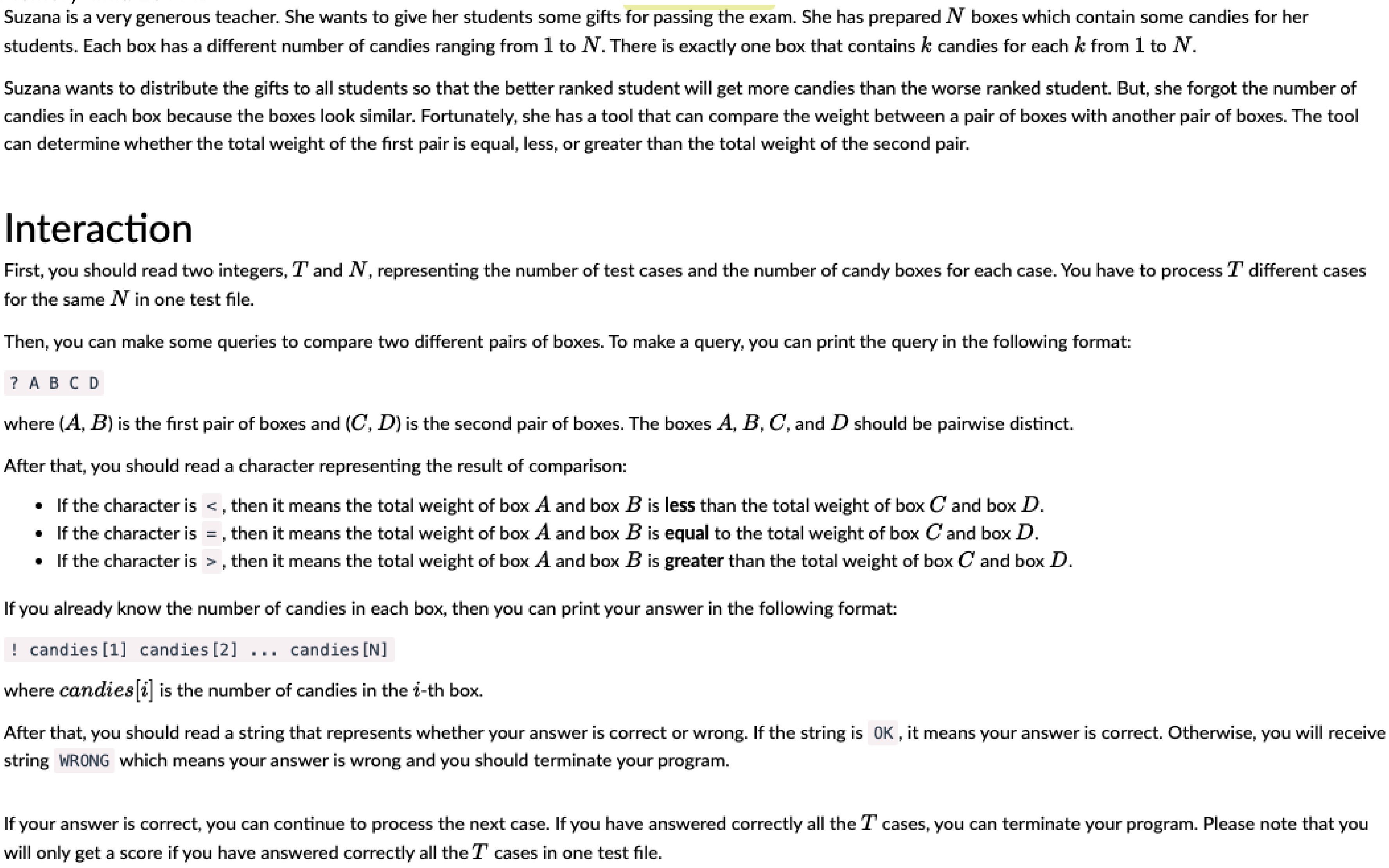


8	5	1	0	4
10	3	4	1	6
4	8	0	0	4
6	4	8	9	3
2	7	5	2	2

Note that the two plants may have different horizontal/vertical directions.

Secret Boxes

Time limit: *3680 ms*
Memory limit: *264 MB*



Suzana is a very generous teacher. She wants to give her students some gifts for passing the exam. She has prepared N boxes which contain some candies for her students. Each box has a different number of candies ranging from 1 to N. There is exactly one box that contains k candies for each k from 1 to N.

Suzana wants to distribute the gifts to all students so that the better ranked student will get more candies than the worse ranked student. But, she forgot the number of candies in each box because the boxes look similar. Fortunately, she has a tool that can compare the weight between a pair of boxes with another pair of boxes. The tool can determine whether the total weight of the first pair is equal, less, or greater than the total weight of the second pair.

Interaction

First, you should read two integers, T and N, representing the number of test cases and the number of candy boxes for each case. You have to process T different cases for the same N in one test file.

Then, you can make some queries to compare two different pairs of boxes. To make a query, you can print the query in the following format:

```
? A B C D
```

where (A, B) is the first pair of boxes and (C, D) is the second pair of boxes. The boxes A, B, C, and D should be pairwise distinct.

After that, you should read a character representing the result of comparison:

- If the character is `<`, then it means the total weight of box A and box B is **less** than the total weight of box C and box D.
- If the character is `=`, then it means the total weight of box A and box B is **equal** to the total weight of box C and box D.
- If the character is `>`, then it means the total weight of box A and box B is **greater** than the total weight of box C and box D.

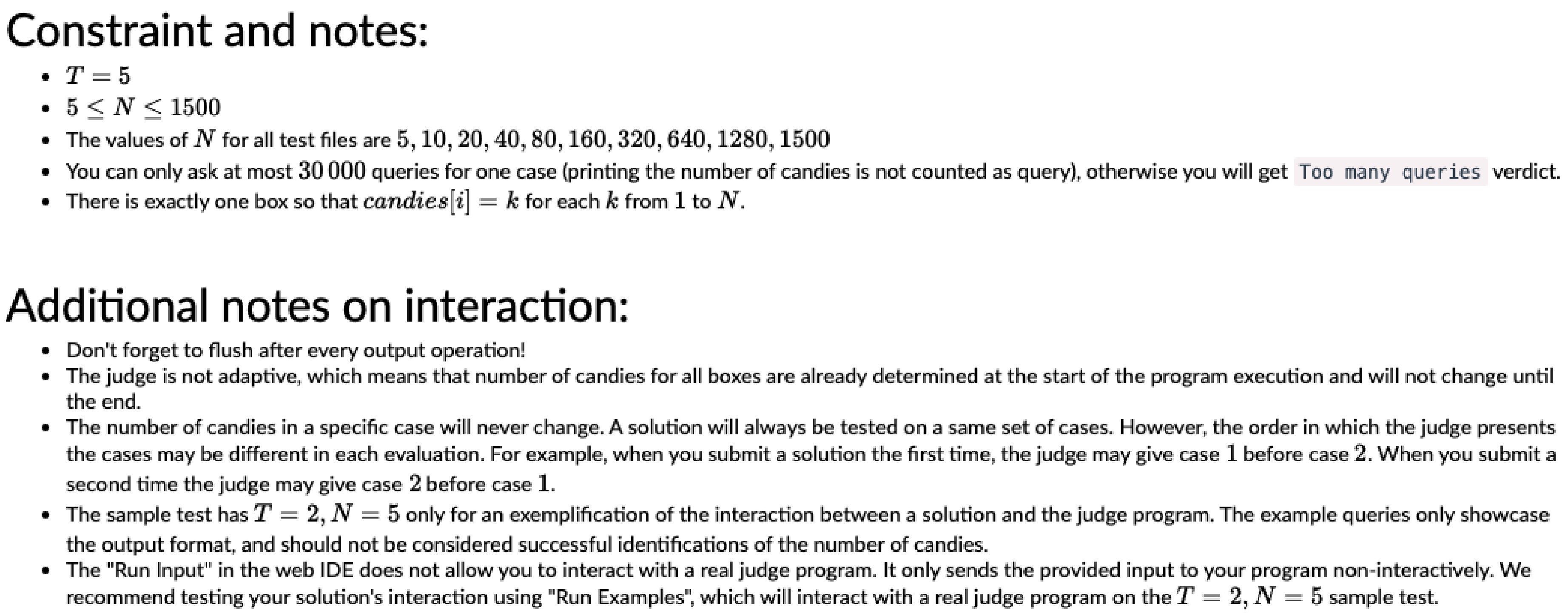
If you already know the number of candies in each box, then you can print your answer in the following format:

```
! candies[1] candies[2] ... candies[N]
```

where $candies[i]$ is the number of candies in the i-th box.

After that, you should read a string that represents whether your answer is correct or wrong. If the string is `OK`, it means your answer is correct. Otherwise, you will receive string `WRONG` which means your answer is wrong and you should terminate your program.

If your answer is correct, you can continue to process the next case. If you have answered correctly all the T cases, you can terminate your program. Please note that you will only get a score if you have answered correctly all the T cases in one test file.

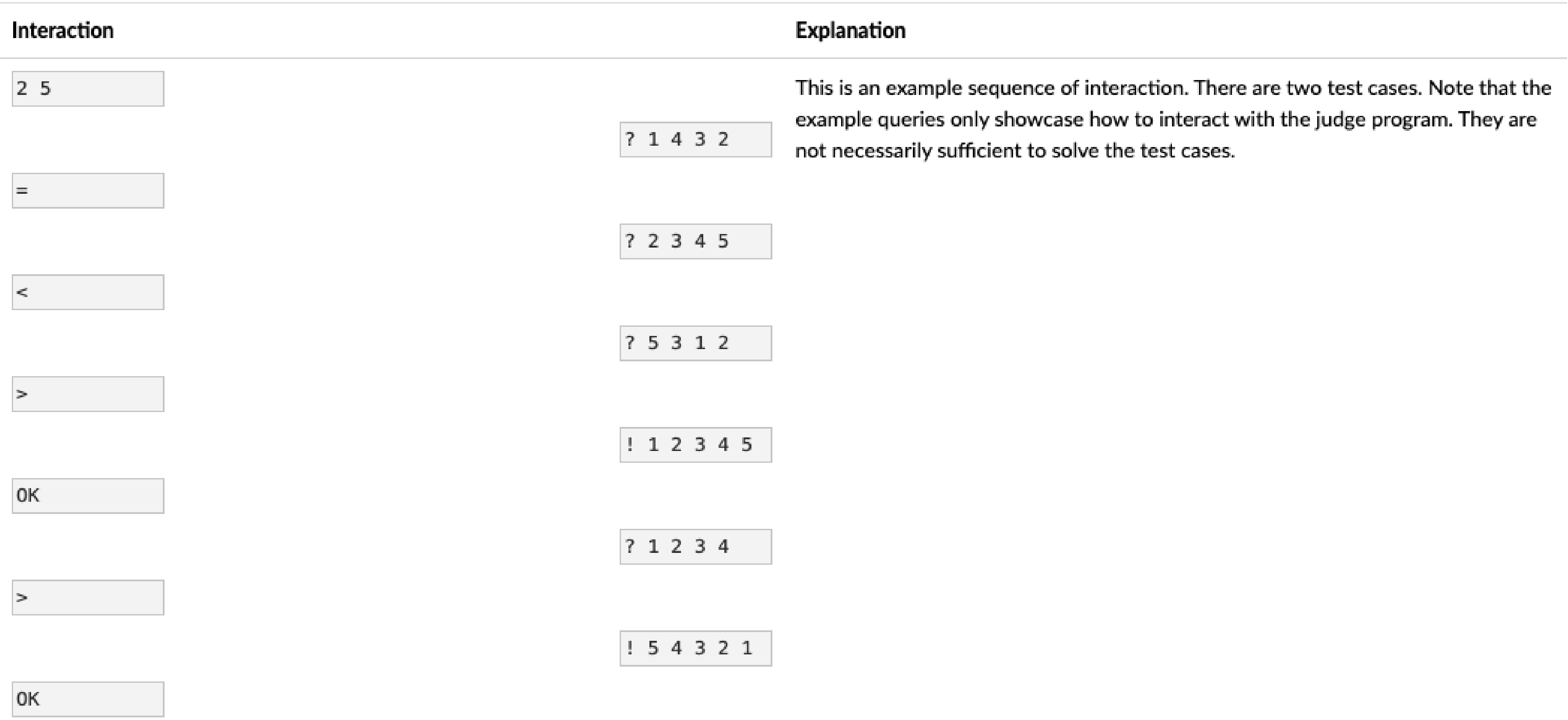


Constraint and notes:

- $T = 5$
- $5 \leq N \leq 1500$
- The values of N for all test files are $5, 10, 20, 40, 80, 160, 320, 640, 1280, 1500$
- You can only ask at most $30\,000$ queries for one case (printing the number of candies is not counted as query), otherwise you will get `Too many queries` verdict.
- There is exactly one box so that $candies[i] = k$ for each k from 1 to N.

Additional notes on interaction:

- Don't forget to flush after every output operation!
- The judge is not adaptive, which means that number of candies for all boxes are already determined at the start of the program execution and will not change until the end.
- The number of candies in a specific case will never change. A solution will always be tested on a same set of cases. However, the order in which the judge presents the cases may be different in each evaluation. For example, when you submit a solution the first time, the judge may give case 1 before case 2. When you submit a second time the judge may give case 2 before case 1.
- The sample test has $T = 2, N = 5$ only for an exemplification of the interaction between a solution and the judge program. The example queries only showcase the output format, and should not be considered successful identifications of the number of candies.
- The "Run Input" in the web IDE does not allow you to interact with a real judge program. It only sends the provided input to your program non-interactively. We recommend testing your solution's interaction using "Run Examples", which will interact with a real judge program on the $T = 2, N = 5$ sample test.



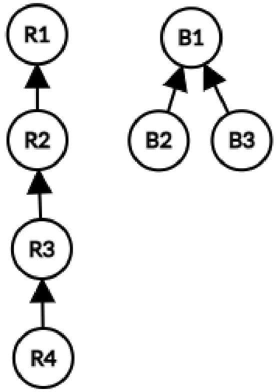
Interaction		Explanation
`2 5`		This is an example sequence of interaction. There are two test cases. Note that the example queries only showcase how to interact with the judge program. They are not necessarily sufficient to solve the test cases.
	`? 1 4 3 2`	
`=`		
	`? 2 3 4 5`	
`<`		
	`? 5 3 1 2`	
`>`		
	`! 1 2 3 4 5`	
`OK`		
	`? 1 2 3 4`	
`>`		
	`! 5 4 3 2 1`	
`OK`		

Spies of Red and Blue

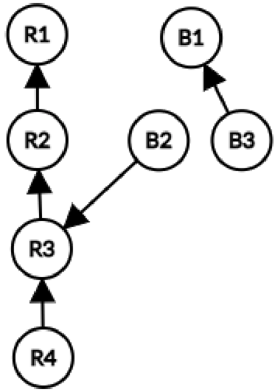
Time limit: *2480 ms*
Memory limit: *264 MB*

Red and Blue are two countries currently engaged in a battle for world dominance in an era of the cold war. Since information is key in winning this war, both countries have invested time, money, and effort in developing spy agencies and have spies across the world to provide them with information in order to act as quickly as possible.

In order to retain secrecy, the spy agencies are structured such that a spy will only report to their immediate senior. Any information received by a spy will be passed to their immediate senior till it reaches the head of the spy agency with R_1 and B_1 being the head of spy agencies of Red and Blue. Spies may be turned over to the other side and the compromised spies will get a new immediate senior to whom he/she will report information.



In the above spy reporting structure, Red has 4 spies and Blue has 3 spies with R_1 and B_1 being the head of their respective countries. It will take 3 interactions for information to reach R_1 from R_4, and it will take 1 interaction for information to reach B_1 from B_2. If B_2 had turned over to the red country with R_3 as his immediate senior, then, the information from B_2 will never reach B_1, and it will reach R_1 via R_3 with 3 interactions instead (as shown below).



Given the spies' organizational structure of Red and Blue countries and the sequence of events as to which spy receives the information or is turned over, find which country will win the information war by getting information faster and how many interactions will it take to pass the information to its head of spy agency.

Standard input

The first line has a single integer T, the number of test cases.

For each test case, the first line consists of three integers N_R, N_B, and E, where N_R is the number of spies in the Red country, N_B is the number of spies in the Blue country, and E is the number of events that will occur. The spies in the Red countries are named $R_1, R_2, \dots R_{N_R}$, and the spies in the Blue countries are named $B_1, B_2, \dots B_{N_B}$.

The second line has $N_R - 1$ space separated strings. The ith of these strings is the name of the immediate senior of spy R_{i+1}.

The third line has $N_B - 1$ space separated strings. The ith of these strings is the name of the immediate senior of spy B_{i+1}.

The next E lines each describe an event. An event is one of the two types:

- `c` x y: There is a spy crossover. Spy x now reports to spy y as his/her immediate senior.
- `w` x y: There is a piece of battle information received by spy x and spy y at the same time.

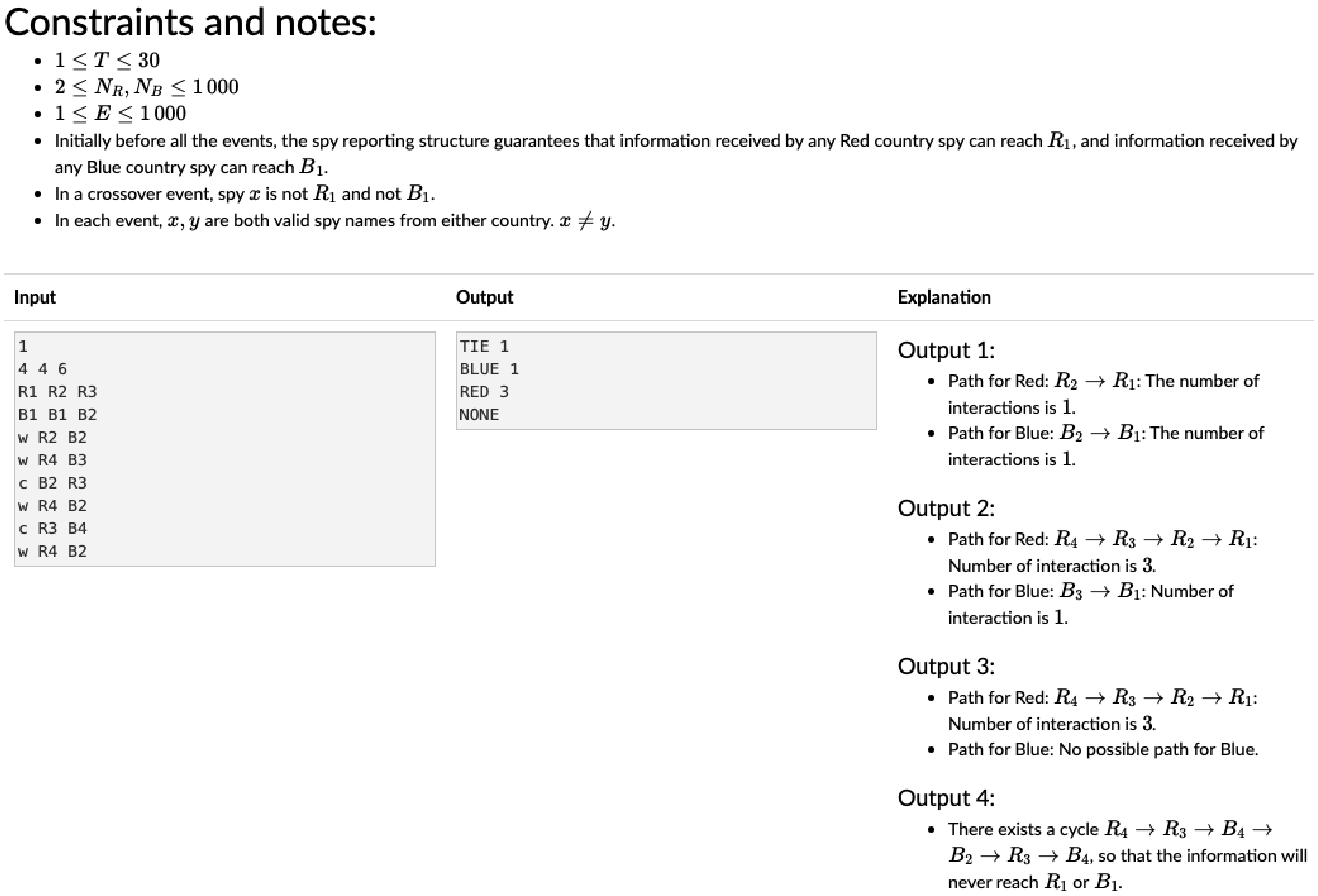
Note that due to the crossovers, x y can be originally spies of a same country. A same crossover may happen multiple times.

Standard output

For each battle information event `w` , output the winner country of the information war, along with k, the minimum number of interactions required for the piece of information to reach its head of spy agency. There can be four possible output scenarios:

- `RED` k: The information reaches R_1 first, in k interactions.
- `BLUE` k: The information reaches B_1 first, in k interactions.
- `TIE` k: The information reaches R_1 and B_1 at the same time, in k interactions.
- `NONE`: The information reaches neither R_1 nor B_1.

Note that k does not need to be printed for the `NONE` scenario.



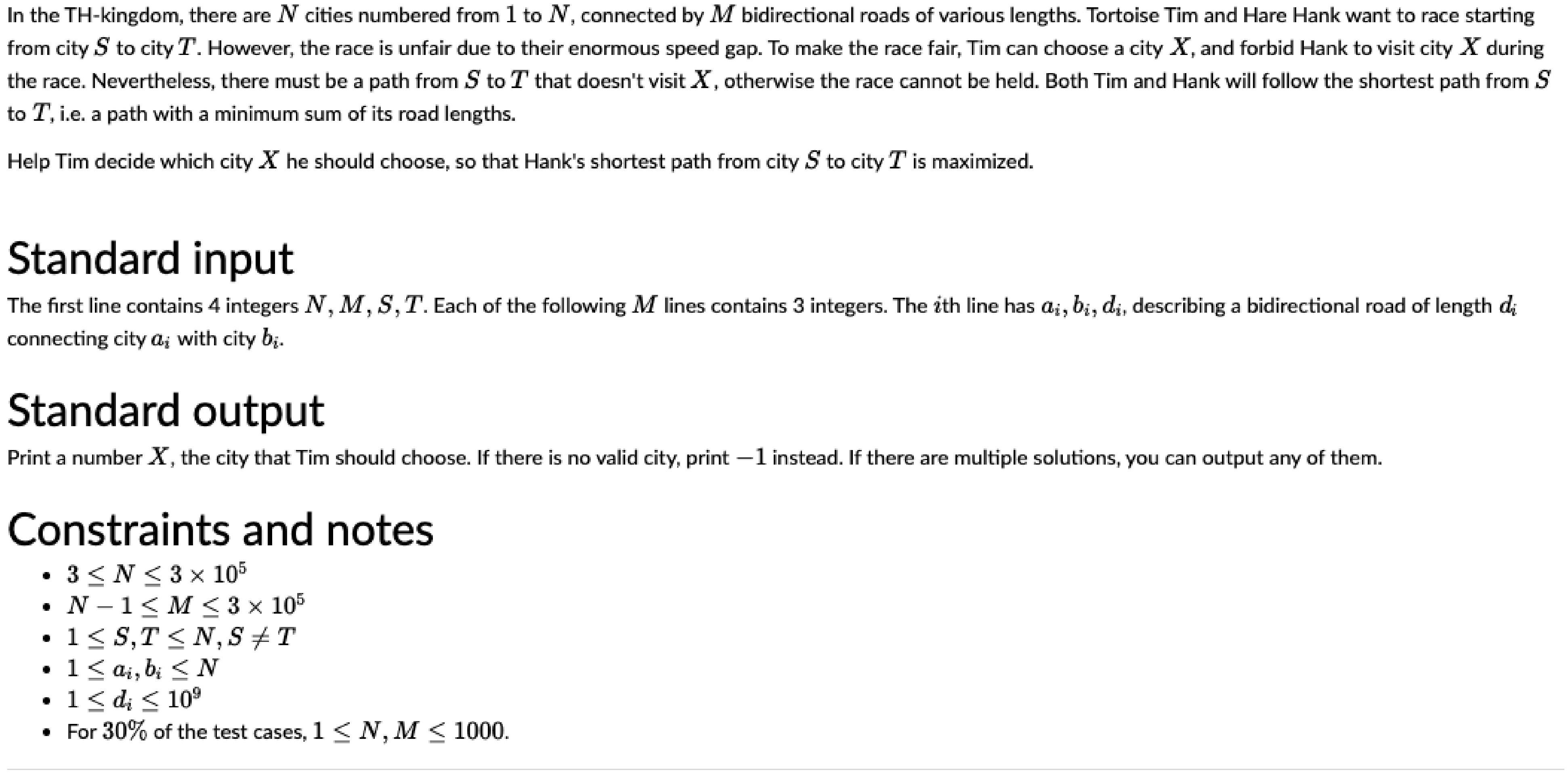
Constraints and notes:

- $1 \leq T \leq 30$
- $2 \leq N_R, N_B \leq 1\,000$
- $1 \leq E \leq 1\,000$
- Initially before all the events, the spy reporting structure guarantees that information received by any Red country spy can reach R_1, and information received by any Blue country spy can reach B_1.
- In a crossover event, spy x is not R_1 and not B_1.
- In each event, x, y are both valid spy names from either country. $x \neq y$.

Input	Output	Explanation
1 4 4 6 R1 R2 R3 B1 B1 B2 w R2 B2 w R4 B3 c B2 R3 w R4 B2 c R3 B4 w R4 B2	TIE 1 BLUE 1 RED 3 NONE	Output 1: • Path for Red: $R_2 \rightarrow R_1$: The number of interactions is 1. • Path for Blue: $B_2 \rightarrow B_1$: The number of interactions is 1. Output 2: • Path for Red: $R_4 \rightarrow R_3 \rightarrow R_2 \rightarrow R_1$: Number of interaction is 3. • Path for Blue: $B_3 \rightarrow B_1$: Number of interaction is 1. Output 3: • Path for Red: $R_4 \rightarrow R_3 \rightarrow R_2 \rightarrow R_1$: Number of interaction is 3. • Path for Blue: No possible path for Blue. Output 4: • There exists a cycle $R_4 \rightarrow R_3 \rightarrow B_4 \rightarrow B_2 \rightarrow R_3 \rightarrow B_4$, so that the information will never reach R_1 or B_1.

The Tortoise and The Hare

Time limit: *2480 ms*
Memory limit: *264 MB*



In the TH-kingdom, there are N cities numbered from 1 to N, connected by M bidirectional roads of various lengths. Tortoise Tim and Hare Hank want to race starting from city S to city T. However, the race is unfair due to their enormous speed gap. To make the race fair, Tim can choose a city X, and forbid Hank to visit city X during the race. Nevertheless, there must be a path from S to T that doesn't visit X, otherwise the race cannot be held. Both Tim and Hank will follow the shortest path from S to T, i.e. a path with a minimum sum of its road lengths.

Help Tim decide which city X he should choose, so that Hank's shortest path from city S to city T is maximized.

Standard input

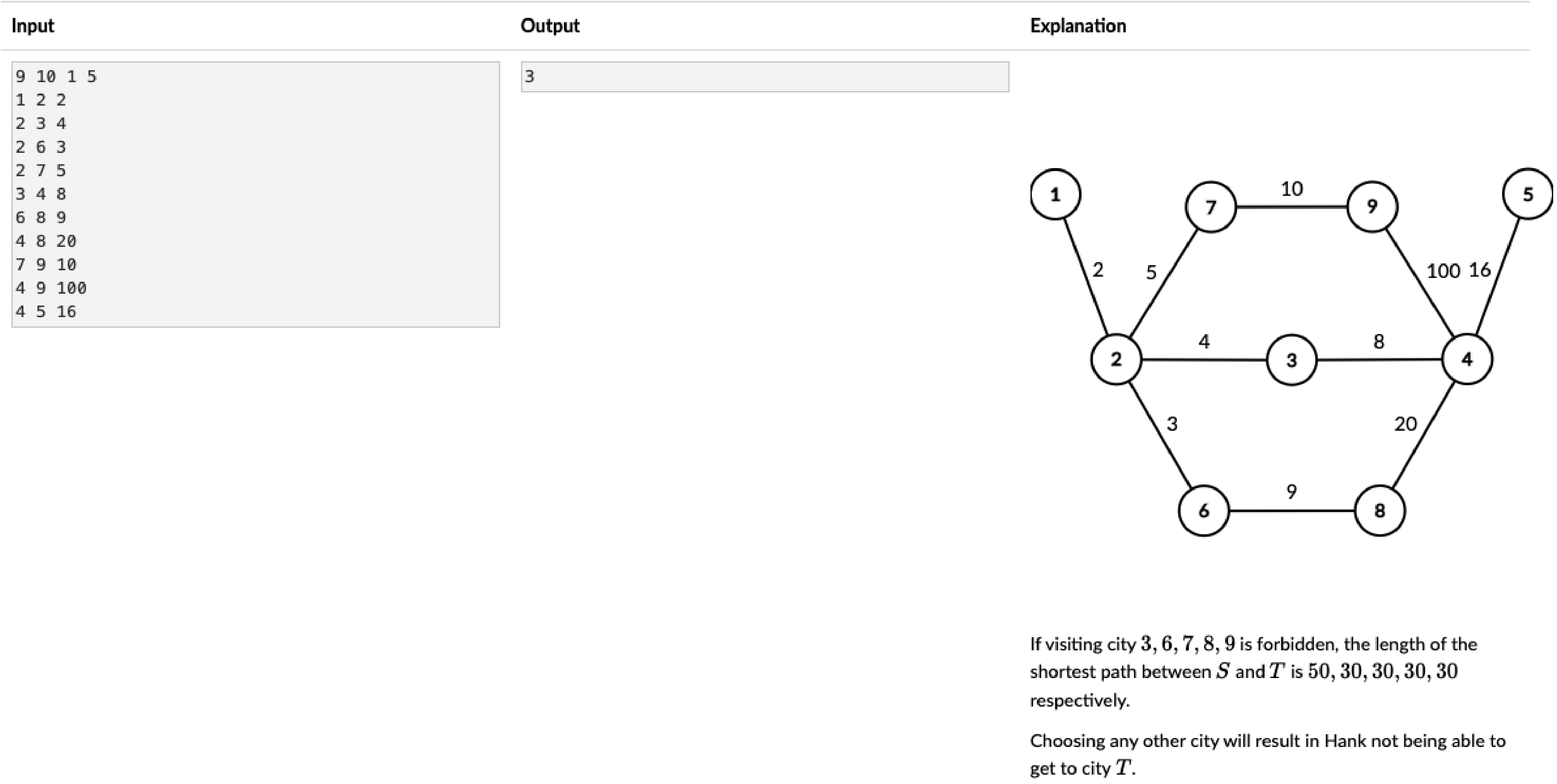
The first line contains 4 integers N, M, S, T. Each of the following M lines contains 3 integers. The ith line has a_i, b_i, d_i, describing a bidirectional road of length d_i connecting city a_i with city b_i.

Standard output

Print a number X, the city that Tim should choose. If there is no valid city, print -1 instead. If there are multiple solutions, you can output any of them.

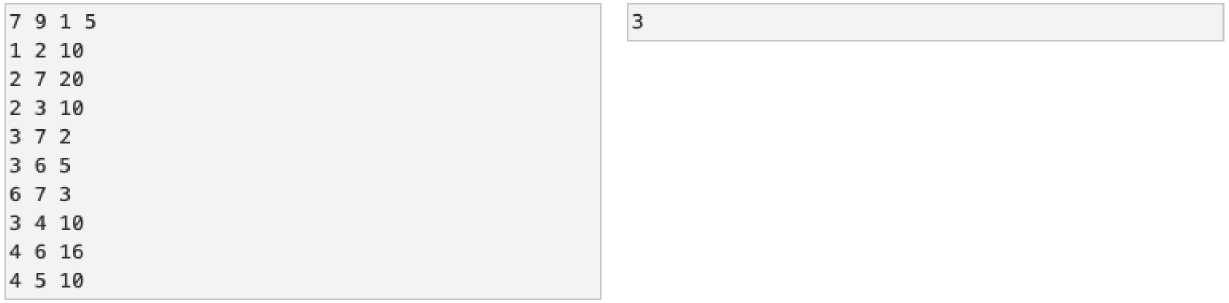
Constraints and notes

- $3 \leq N \leq 3 \times 10^5$
- $N - 1 \leq M \leq 3 \times 10^5$
- $1 \leq S, T \leq N, S \neq T$
- $1 \leq a_i, b_i \leq N$
- $1 \leq d_i \leq 10^9$
- For 30% of the test cases, $1 \leq N, M \leq 1000$.



Input	Output	Explanation

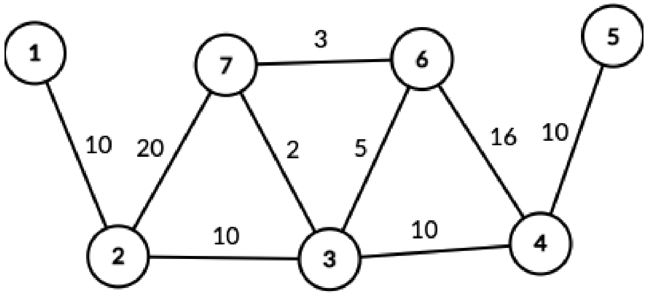
9 10 1 5
1 2 2
2 3 4
2 6 3
2 7 5
3 4 8
6 8 9
4 8 20
7 9 10
4 9 100
4 5 16
	3	If visiting city $3, 6, 7, 8, 9$ is forbidden, the length of the shortest path between S and T is $50, 30, 30, 30, 30$ respectively.
		Choosing any other city will result in Hank not being able to get to city T.



```
7 9 1 5
1 2 10
2 7 20
2 3 10
3 7 2
3 6 5
6 7 3
3 4 10
4 6 16
4 5 10
```

```
3
```

The cities and roads are illustrated below:

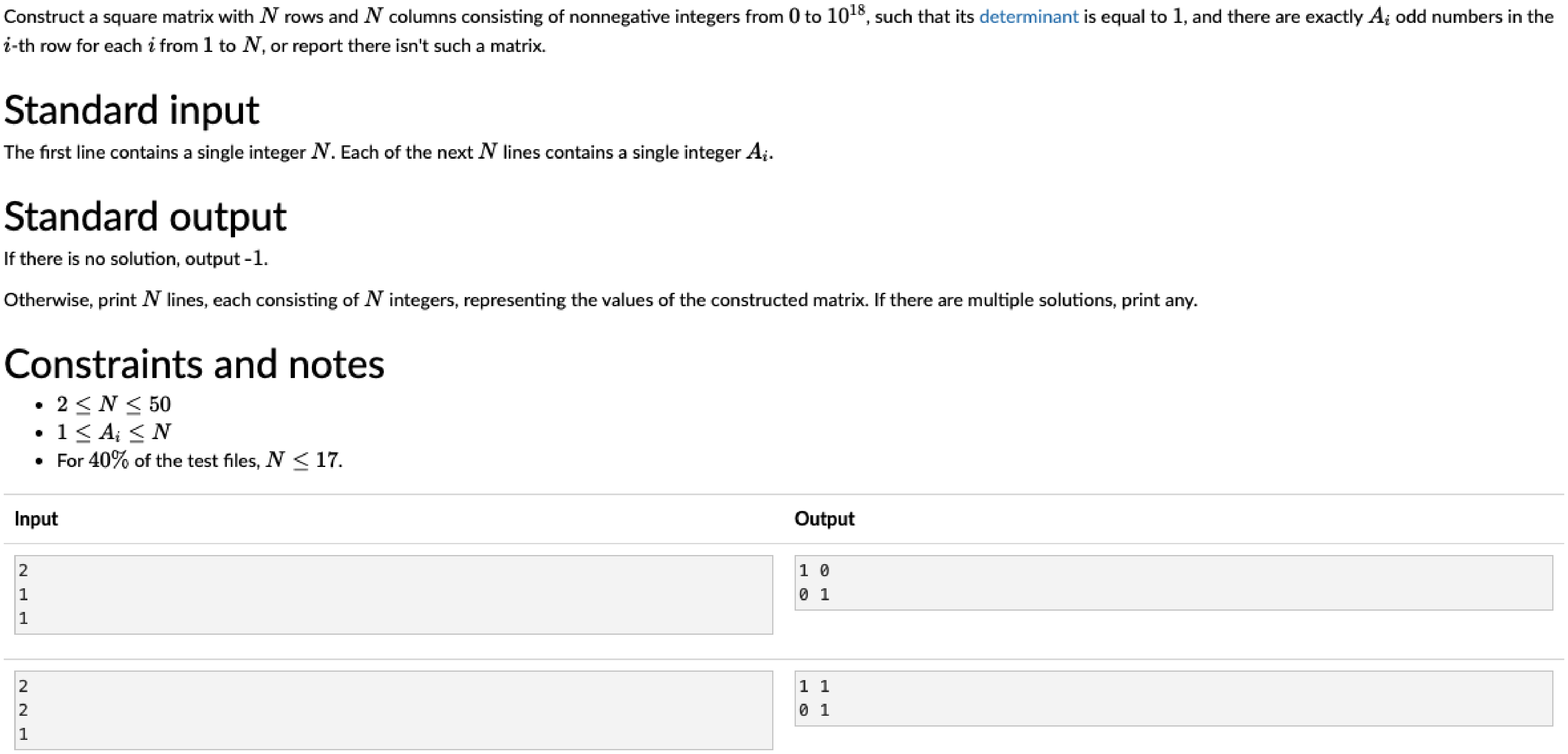


If visiting city $3, 6, 7$ is forbidden, the length of the shortest path between S and T is $59, 40, 40$ respectively.

Choosing any other city will result in Hank not being able to get to city T.

Big Matrix Small Determinant

Time limit: *1280 ms*
Memory limit: *264 MB*



Construct a square matrix with N rows and N columns consisting of nonnegative integers from 0 to 10^{18}, such that its determinant is equal to 1, and there are exactly A_i odd numbers in the i-th row for each i from 1 to N, or report there isn't such a matrix.

Standard input

The first line contains a single integer N. Each of the next N lines contains a single integer A_i.

Standard output

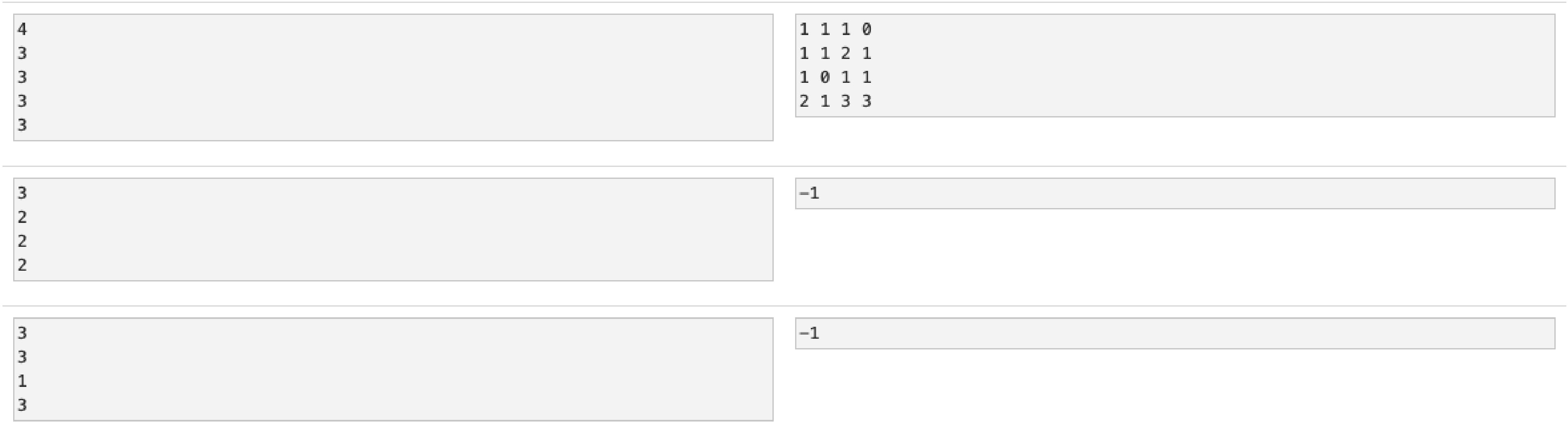
If there is no solution, output -1.

Otherwise, print N lines, each consisting of N integers, representing the values of the constructed matrix. If there are multiple solutions, print any.

Constraints and notes

- $2 \leq N \leq 50$
- $1 \leq A_i \leq N$
- For 40% of the test files, $N \leq 17$.

Input	**Output**
2 1 1	1 0 0 1
2 2 1	1 1 0 1



```
4
3
3
3
3
```

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

```
3
2
2
2
```

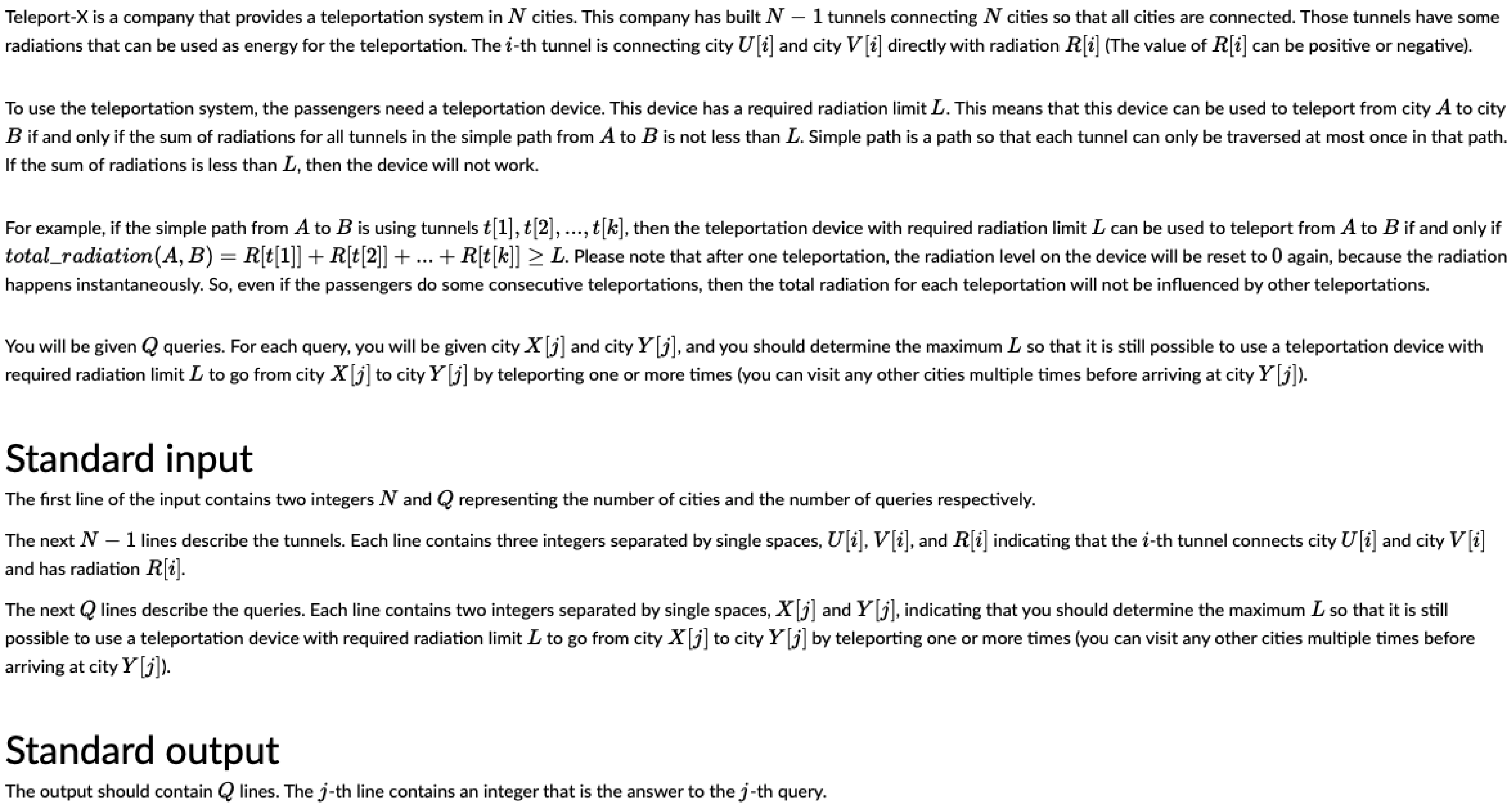
```
-1
```

```
3
3
1
3
```

```
-1
```

Xtreme Teleportation

Time limit: *7280 ms*
Memory limit: *264 MB*



Teleport-X is a company that provides a teleportation system in N cities. This company has built $N-1$ tunnels connecting N cities so that all cities are connected. Those tunnels have some radiations that can be used as energy for the teleportation. The i-th tunnel is connecting city $U[i]$ and city $V[i]$ directly with radiation $R[i]$ (The value of $R[i]$ can be positive or negative).

To use the teleportation system, the passengers need a teleportation device. This device has a required radiation limit L. This means that this device can be used to teleport from city A to city B if and only if the sum of radiations for all tunnels in the simple path from A to B is not less than L. Simple path is a path so that each tunnel can only be traversed at most once in that path. If the sum of radiations is less than L, then the device will not work.

For example, if the simple path from A to B is using tunnels $t[1], t[2], ..., t[k]$, then the teleportation device with required radiation limit L can be used to teleport from A to B if and only if $total\_radiation(A, B) = R[t[1]] + R[t[2]] + ... + R[t[k]] \geq L$. Please note that after one teleportation, the radiation level on the device will be reset to 0 again, because the radiation happens instantaneously. So, even if the passengers do some consecutive teleportations, then the total radiation for each teleportation will not be influenced by other teleportations.

You will be given Q queries. For each query, you will be given city $X[j]$ and city $Y[j]$, and you should determine the maximum L so that it is still possible to use a teleportation device with required radiation limit L to go from city $X[j]$ to city $Y[j]$ by teleporting one or more times (you can visit any other cities multiple times before arriving at city $Y[j]$).

Standard input

The first line of the input contains two integers N and Q representing the number of cities and the number of queries respectively.

The next $N-1$ lines describe the tunnels. Each line contains three integers separated by single spaces, $U[i]$, $V[i]$, and $R[i]$ indicating that the i-th tunnel connects city $U[i]$ and city $V[i]$ and has radiation $R[i]$.

The next Q lines describe the queries. Each line contains two integers separated by single spaces, $X[j]$ and $Y[j]$, indicating that you should determine the maximum L so that it is still possible to use a teleportation device with required radiation limit L to go from city $X[j]$ to city $Y[j]$ by teleporting one or more times (you can visit any other cities multiple times before arriving at city $Y[j]$).

Standard output

The output should contain Q lines. The j-th line contains an integer that is the answer to the j-th query.

Constraints and notes

- $2 \le N \le 100\,000$
- $1 \le U[i], V[i] \le N$
- $U[i] \ne V[i]$
- $-10^9 \le R[i] \le 10^9$
- $1 \le Q \le 200\,000$
- $1 \le X[j], Y[j] \le N$
- $X[j] \ne Y[j]$

- For 25% of the test data, $N, Q \le 100$.
- For 50% of the test data, $N, Q \le 1\,000$.

Input

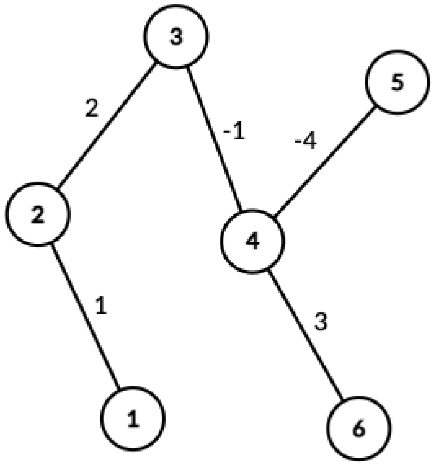
```
6 5
1 2 1
2 3 2
3 4 -1
4 5 -4
4 6 3
1 3
2 6
6 3
4 5
3 2
```

Output

```
3
4
3
-1
3
```

Explanation

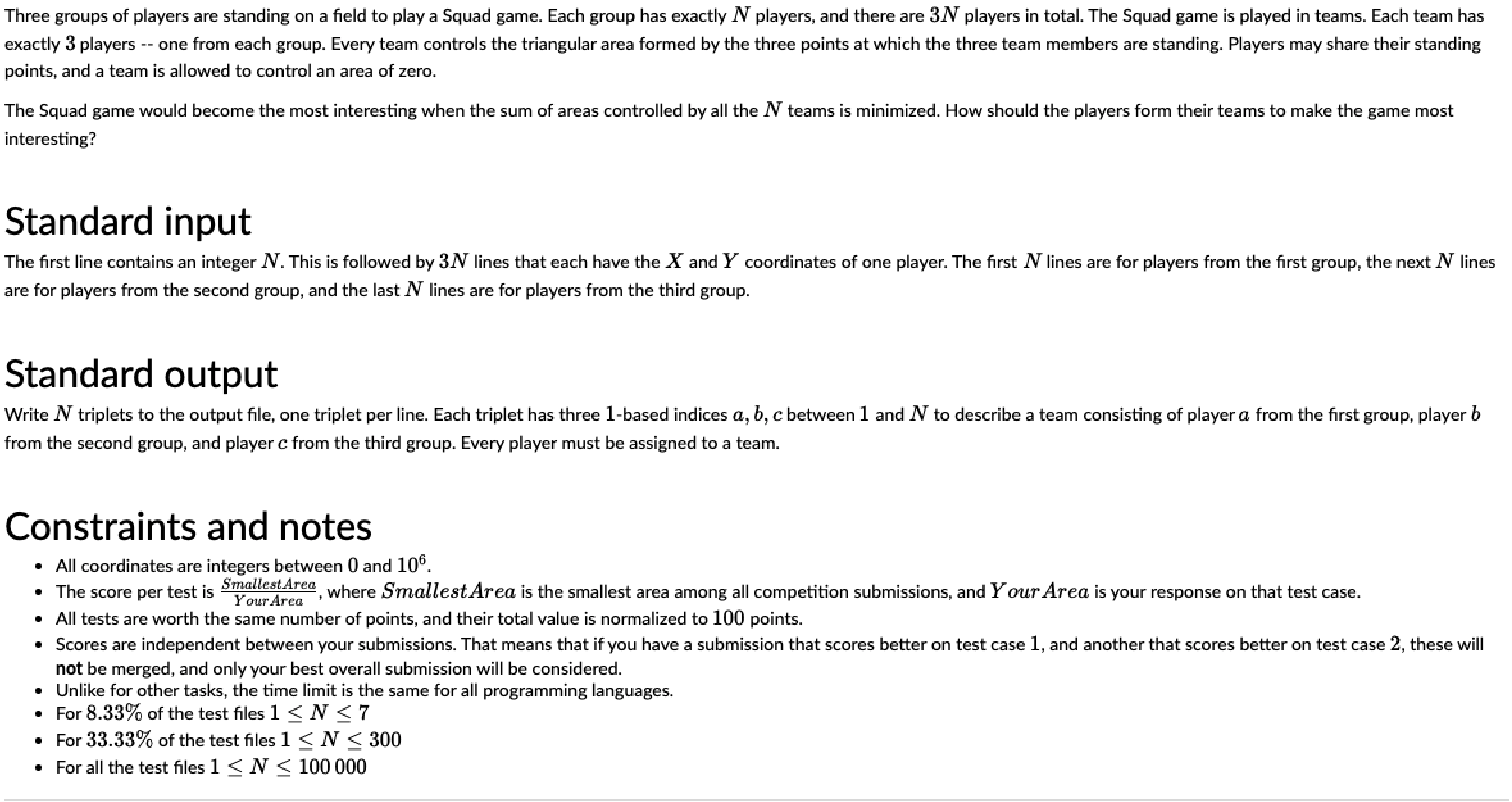
Below is the illustration of the tunnels.



- In the first query, we can teleport from city 1 to city 3 directly with $total\_radiation(1, 3) = 3$. So, we can use a device with $L = 3$. There is no way we can reach city 3 from city 1 if $L > 3$.
- In the second query, the maximum L is 4. We can also teleport from city 2 to city 6 directly with $total\_radiation(2, 6) = 4$.
- In the third query, the maximum L is 3. First, We can teleport from city 6 to city 1 with $total\_radiation(6, 1) = 5$. And then, we can teleport from city 1 to city 3 with $total\_radiation(1, 3) = 3$.
- In the fourth query, the maximum L is -1. First, We can teleport from city 4 to city 6 with $total\_radiation(4, 6) = 3$. And then, we can teleport from city 6 to city 5 with $total\_radiation(6, 5) = -1$.
- In the fifth query, the maximum L is 3. First, we can teleport from city 3 to city 1 with $total\_radiation(3, 1) = 3$. And then, we can teleport from city 1 to city 6 with $total\_radiation(1, 6) = 5$. And finally, we can teleport from city 6 to city 2 with $total\_radiation(6, 2) = 4$.

Squad Game

Time limit: *2080 ms*
Memory limit: *1 GB*



Three groups of players are standing on a field to play a Squad game. Each group has exactly N players, and there are $3N$ players in total. The Squad game is played in teams. Each team has exactly 3 players -- one from each group. Every team controls the triangular area formed by the three points at which the three team members are standing. Players may share their standing points, and a team is allowed to control an area of zero.

The Squad game would become the most interesting when the sum of areas controlled by all the N teams is minimized. How should the players form their teams to make the game most interesting?

Standard input

The first line contains an integer N. This is followed by $3N$ lines that each have the X and Y coordinates of one player. The first N lines are for players from the first group, the next N lines are for players from the second group, and the last N lines are for players from the third group.

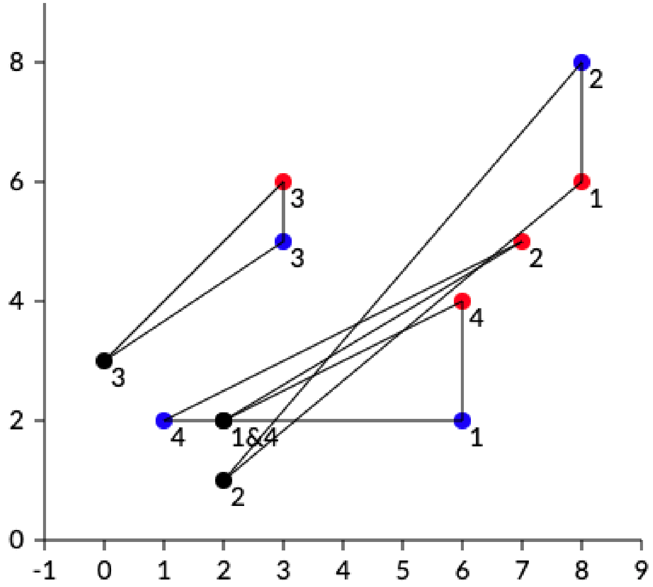
Standard output

Write N triplets to the output file, one triplet per line. Each triplet has three 1-based indices a, b, c between 1 and N to describe a team consisting of player a from the first group, player b from the second group, and player c from the third group. Every player must be assigned to a team.

Constraints and notes

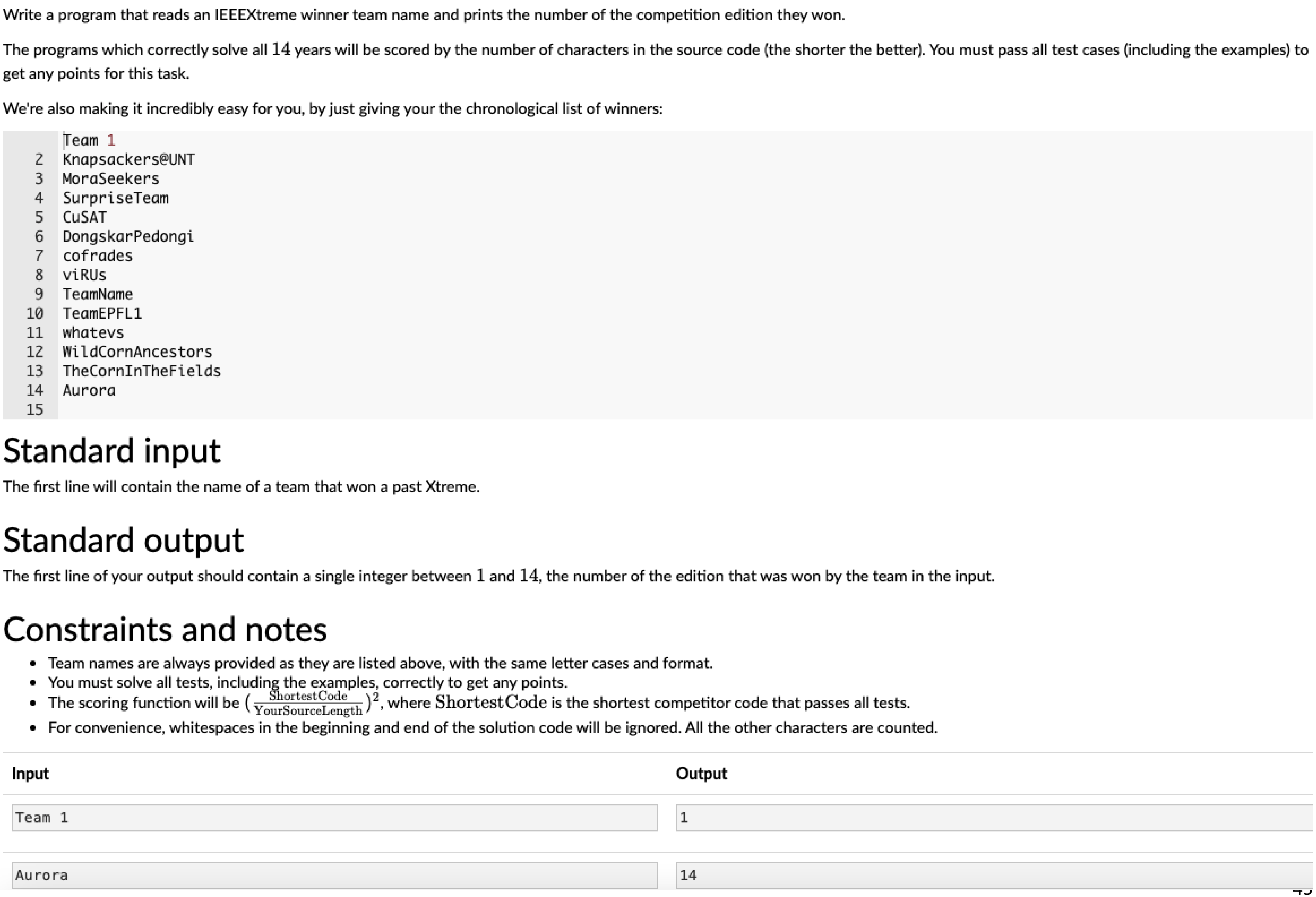
- All coordinates are integers between 0 and 10^6.
- The score per test is $\frac{SmallestArea}{YourArea}$, where $SmallestArea$ is the smallest area among all competition submissions, and $YourArea$ is your response on that test case.
- All tests are worth the same number of points, and their total value is normalized to 100 points.
- Scores are independent between your submissions. That means that if you have a submission that scores better on test case 1, and another that scores better on test case 2, these will **not** be merged, and only your best overall submission will be considered.
- Unlike for other tasks, the time limit is the same for all programming languages.
- For 8.33% of the test files $1 \leq N \leq 7$
- For 33.33% of the test files $1 \leq N \leq 300$
- For all the test files $1 \leq N \leq 100\,000$

Input	Output	Explanation

4
6 2
8 8
3 5
1 2
8 6
7 5
3 6
6 4
2 2
2 1
0 3
2 2
	1 4 1
	2 1 2
	3 3 3
	4 2 4
		In the figure below, players in the first group are marked in blue, the second group in red and the third group in black: 

Xtreme Winners

Time limit: *680 ms*
Memory limit: *264 MB*



Write a program that reads an IEEEXtreme winner team name and prints the number of the competition edition they won.

The programs which correctly solve all 14 years will be scored by the number of characters in the source code (the shorter the better). You must pass all test cases (including the examples) to get any points for this task.

We're also making it incredibly easy for you, by just giving your the chronological list of winners:

```
Team 1
Knapsackers@UNT
MoraSeekers
SurpriseTeam
CuSAT
DongskarPedongi
cofrades
viRUs
TeamName
TeamEPFL1
whatevs
WildCornAncestors
TheCornInTheFields
Aurora

```

Standard input

The first line will contain the name of a team that won a past Xtreme.

Standard output

The first line of your output should contain a single integer between 1 and 14, the number of the edition that was won by the team in the input.

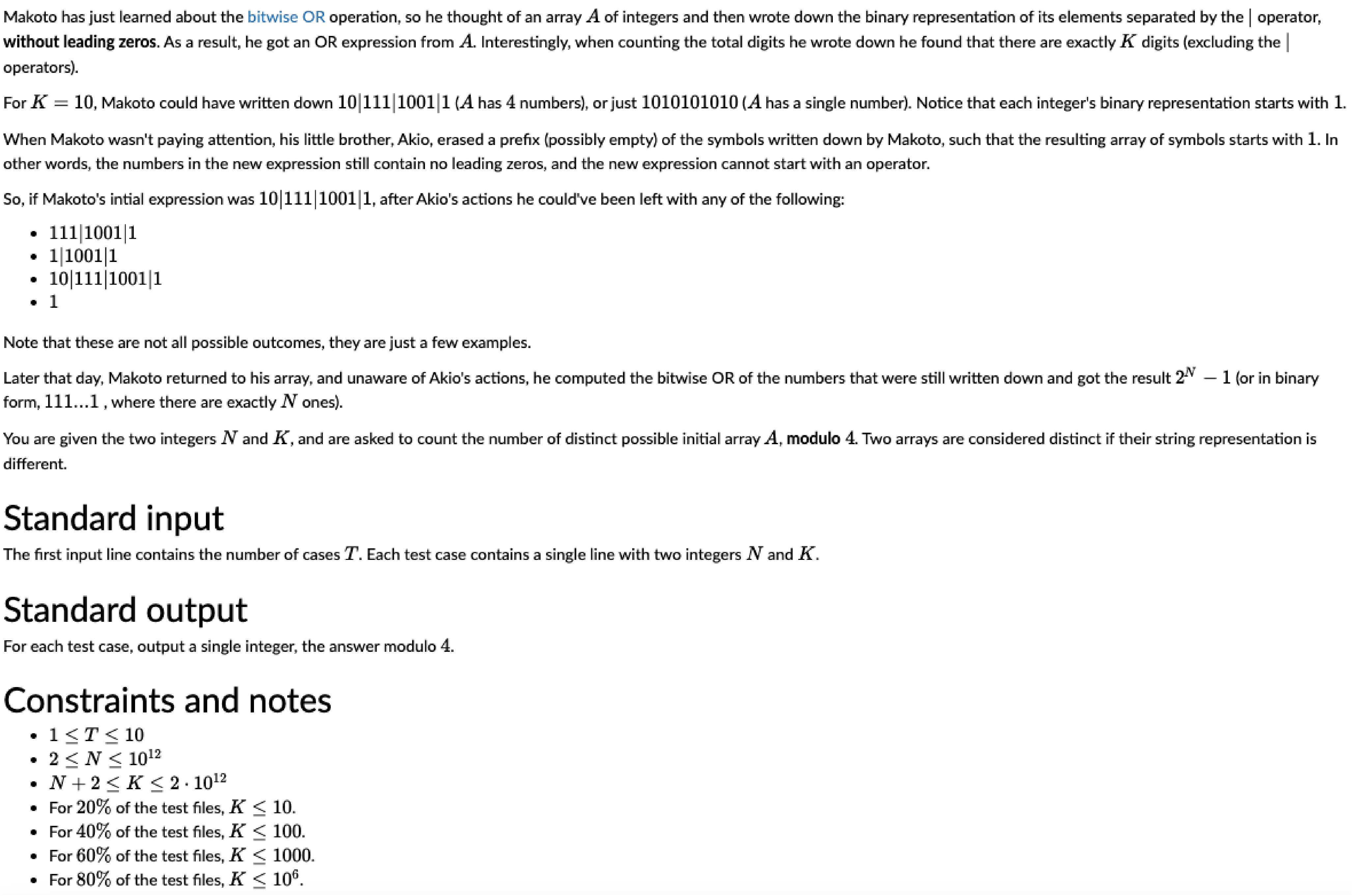
Constraints and notes

- Team names are always provided as they are listed above, with the same letter cases and format.
- You must solve all tests, including the examples, correctly to get any points.
- The scoring function will be $(\frac{\text{ShortestCode}}{\text{YourSourceLength}})^2$, where ShortestCode is the shortest competitor code that passes all tests.
- For convenience, whitespaces in the beginning and end of the solution code will be ignored. All the other characters are counted.

Input	Output
Team 1	1
Aurora	14

Arrays Count

Time limit: *2480 ms*
Memory limit: *264 MB*



Makoto has just learned about the bitwise OR operation, so he thought of an array A of integers and then wrote down the binary representation of its elements separated by the $|$ operator, **without leading zeros**. As a result, he got an OR expression from A. Interestingly, when counting the total digits he wrote down he found that there are exactly K digits (excluding the $|$ operators).

For $K = 10$, Makoto could have written down $10|111|1001|1$ (A has 4 numbers), or just 1010101010 (A has a single number). Notice that each integer's binary representation starts with 1.

When Makoto wasn't paying attention, his little brother, Akio, erased a prefix (possibly empty) of the symbols written down by Makoto, such that the resulting array of symbols starts with 1. In other words, the numbers in the new expression still contain no leading zeros, and the new expression cannot start with an operator.

So, if Makoto's intial expression was $10|111|1001|1$, after Akio's actions he could've been left with any of the following:

- $111|1001|1$
- $1|1001|1$
- $10|111|1001|1$
- 1

Note that these are not all possible outcomes, they are just a few examples.

Later that day, Makoto returned to his array, and unaware of Akio's actions, he computed the bitwise OR of the numbers that were still written down and got the result $2^N - 1$ (or in binary form, $111...1$, where there are exactly N ones).

You are given the two integers N and K, and are asked to count the number of distinct possible initial array A, **modulo** 4. Two arrays are considered distinct if their string representation is different.

Standard input

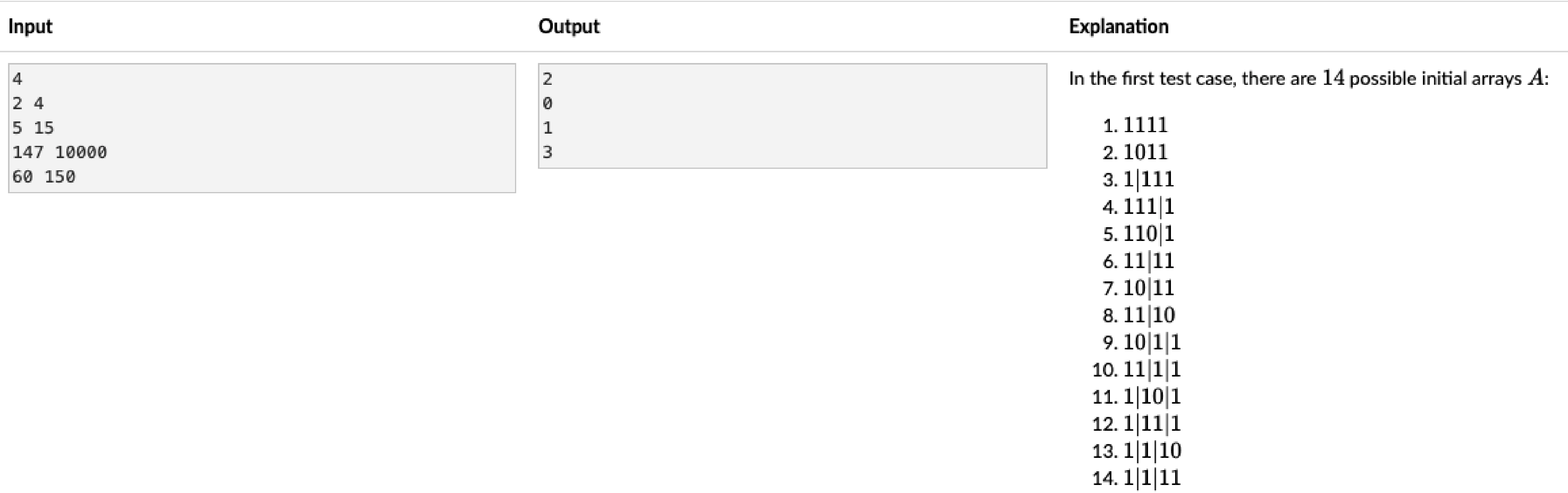
The first input line contains the number of cases T. Each test case contains a single line with two integers N and K.

Standard output

For each test case, output a single integer, the answer modulo 4.

Constraints and notes

- $1 \leq T \leq 10$
- $2 \leq N \leq 10^{12}$
- $N + 2 \leq K \leq 2 \cdot 10^{12}$
- For 20% of the test files, $K \leq 10$.
- For 40% of the test files, $K \leq 100$.
- For 60% of the test files, $K \leq 1000$.
- For 80% of the test files, $K \leq 10^6$.

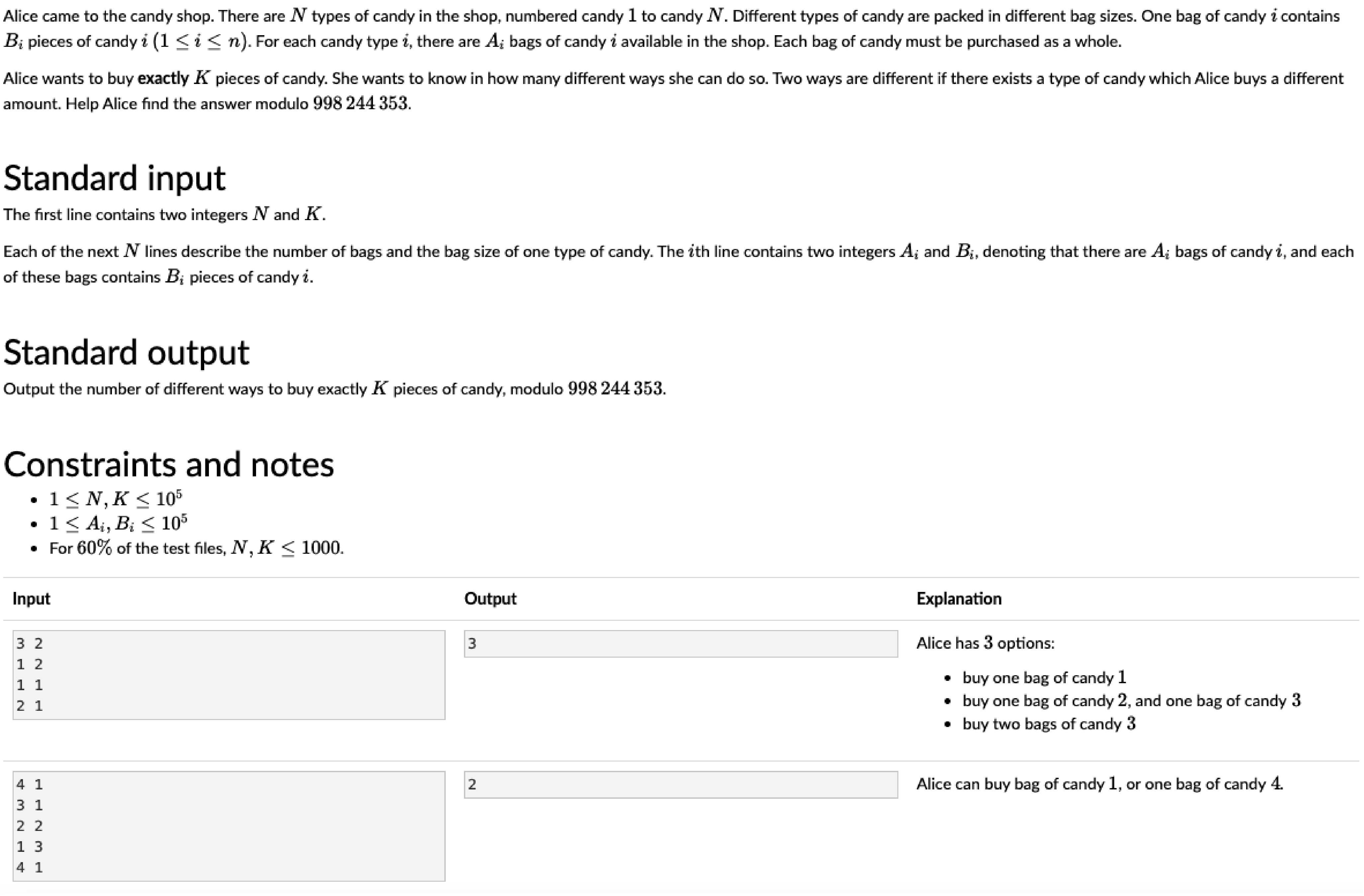


Input	Output	Explanation

4
2 4
5 15
147 10000
60 150
	2
	0
	1
	3
		In the first test case, there are 14 possible initial arrays A:
		1. 1111
		2. 1011
		3. $1\|111$
		4. $111\|1$
		5. $110\|1$
		6. $11\|11$
		7. $10\|11$
		8. $11\|10$
		9. $10\|1\|1$
		10. $11\|1\|1$
		11. $1\|10\|1$
		12. $1\|11\|1$
		13. $1\|1\|10$
		14. $1\|1\|11$

Candy Shop

Time limit: *3680 ms*
Memory limit: *264 MB*



Alice came to the candy shop. There are N types of candy in the shop, numbered candy 1 to candy N. Different types of candy are packed in different bag sizes. One bag of candy i contains B_i pieces of candy i $(1 \le i \le n)$. For each candy type i, there are A_i bags of candy i available in the shop. Each bag of candy must be purchased as a whole.

Alice wants to buy **exactly** K pieces of candy. She wants to know in how many different ways she can do so. Two ways are different if there exists a type of candy which Alice buys a different amount. Help Alice find the answer modulo $998\,244\,353$.

Standard input

The first line contains two integers N and K.

Each of the next N lines describe the number of bags and the bag size of one type of candy. The ith line contains two integers A_i and B_i, denoting that there are A_i bags of candy i, and each of these bags contains B_i pieces of candy i.

Standard output

Output the number of different ways to buy exactly K pieces of candy, modulo $998\,244\,353$.

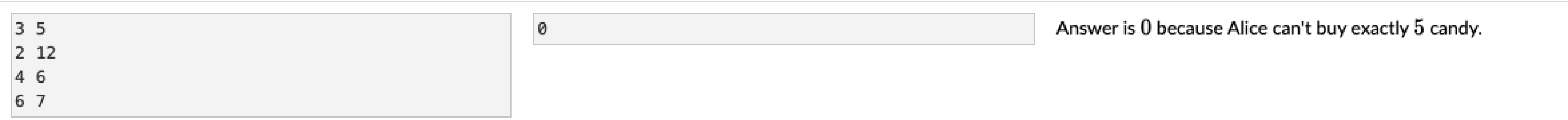
Constraints and notes

- $1 \le N, K \le 10^5$
- $1 \le A_i, B_i \le 10^5$
- For 60% of the test files, $N, K \le 1000$.

Input	Output	Explanation

3 2
1 2
1 1
2 1
	3	Alice has 3 options:
		• buy one bag of candy 1
		• buy one bag of candy 2, and one bag of candy 3
		• buy two bags of candy 3

4 1
3 1
2 2
1 3
4 1
	2	Alice can buy bag of candy 1, or one bag of candy 4.



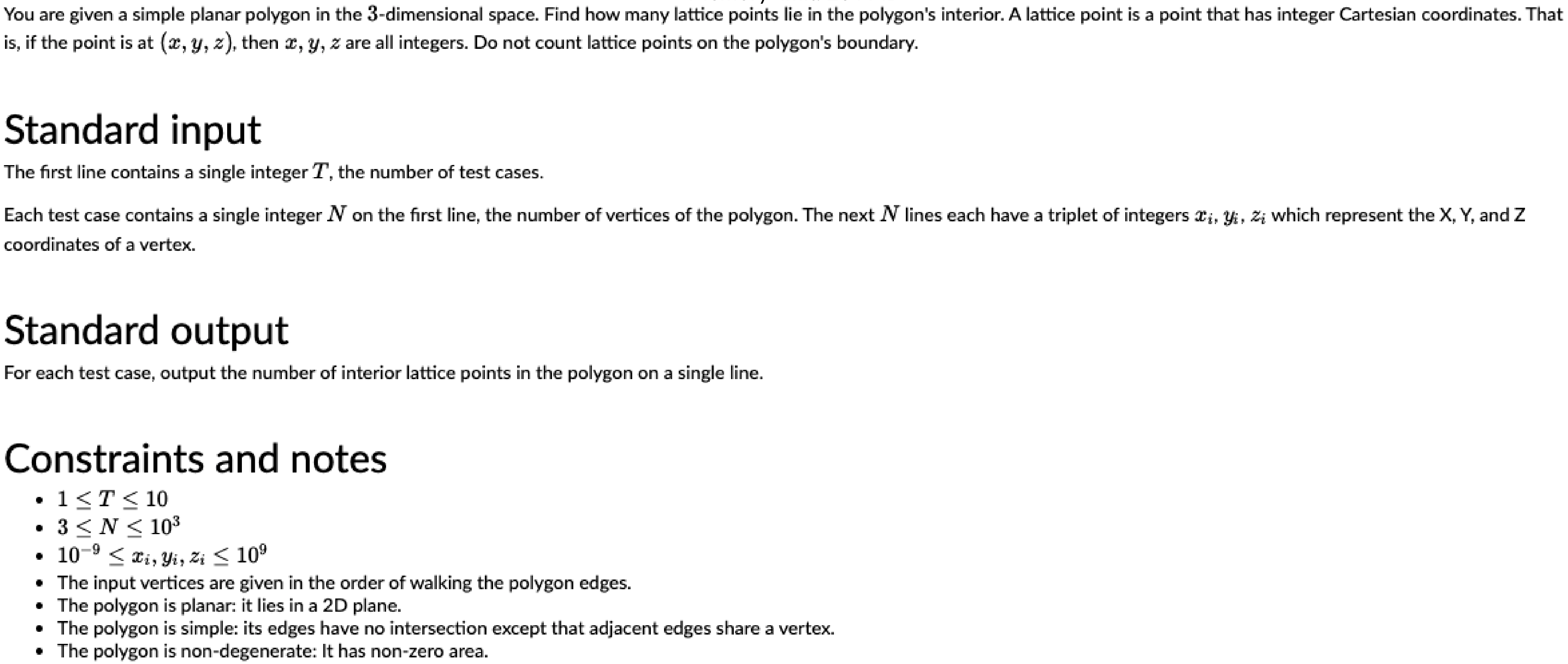
```
3 5
2 12
4 6
6 7
```

```
0
```

Answer is 0 because Alice can't buy exactly 5 candy.

Polygon

Time limit: *1280 ms*
Memory limit: *264 MB*



You are given a simple planar polygon in the 3-dimensional space. Find how many lattice points lie in the polygon's interior. A lattice point is a point that has integer Cartesian coordinates. That is, if the point is at (x, y, z), then x, y, z are all integers. Do not count lattice points on the polygon's boundary.

Standard input

The first line contains a single integer T, the number of test cases.

Each test case contains a single integer N on the first line, the number of vertices of the polygon. The next N lines each have a triplet of integers x_i, y_i, z_i which represent the X, Y, and Z coordinates of a vertex.

Standard output

For each test case, output the number of interior lattice points in the polygon on a single line.

Constraints and notes

- $1 \leq T \leq 10$
- $3 \leq N \leq 10^3$
- $10^{-9} \leq x_i, y_i, z_i \leq 10^9$
- The input vertices are given in the order of walking the polygon edges.
- The polygon is planar: it lies in a 2D plane.
- The polygon is simple: its edges have no intersection except that adjacent edges share a vertex.
- The polygon is non-degenerate: It has non-zero area.

Input

```
3
3
0 0 0
0 1 0
1 0 0
4
-10 44 10
-10 44 11
-11 44 11
-11 44 10
4
4 4 3
12 4 9
12 1 9
4 1 3
```

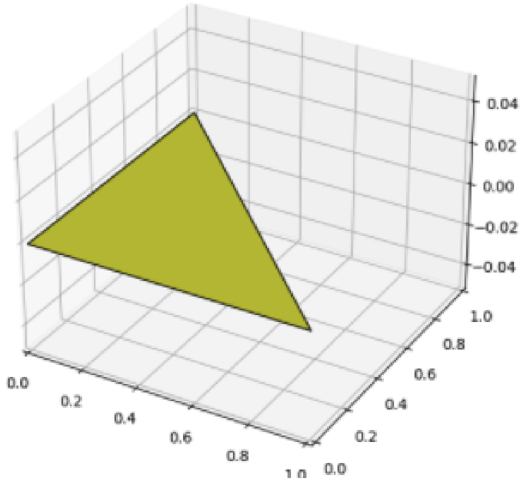
Output

```
0
0
2
```

Explanation

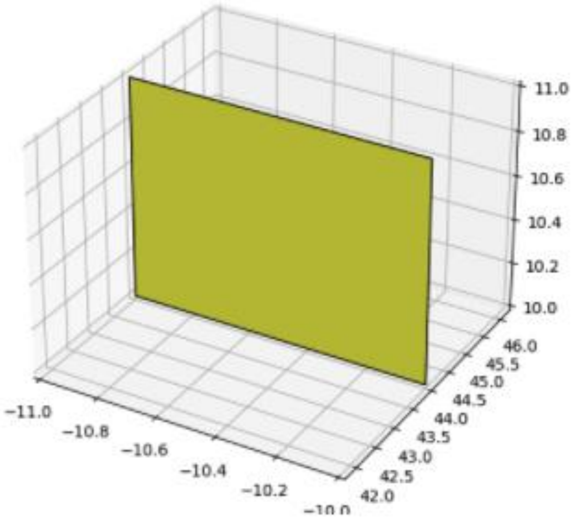
There are three test cases:

- Case 1: The polygon is a smallest right triangle in XY plane, which does not have any interior lattice point.

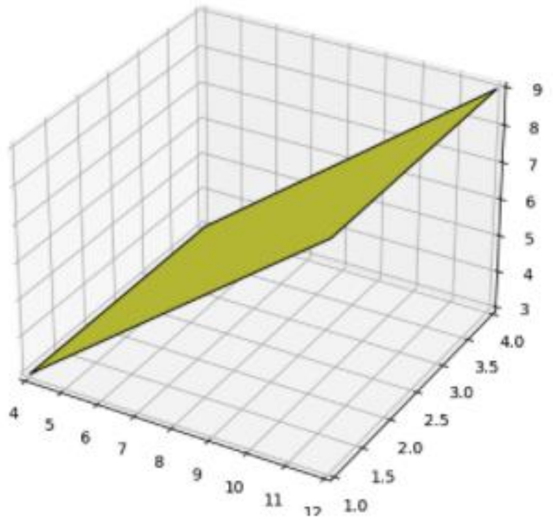


- Case 2: The polygon is a smallest square in a plane parallel to the XZ plane, which also does not have any interior lattice point.

- Case 2: The polygon is a smallest square in a plane parallel to the XZ plane, which also does not have any interior lattice point.



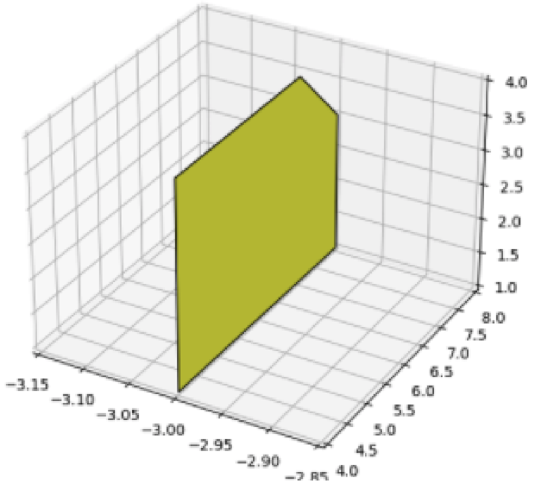
- Case 3: The polygon is a 3×10 rectangle on an inclined plane. It has 2 (2×1) interior lattice points.



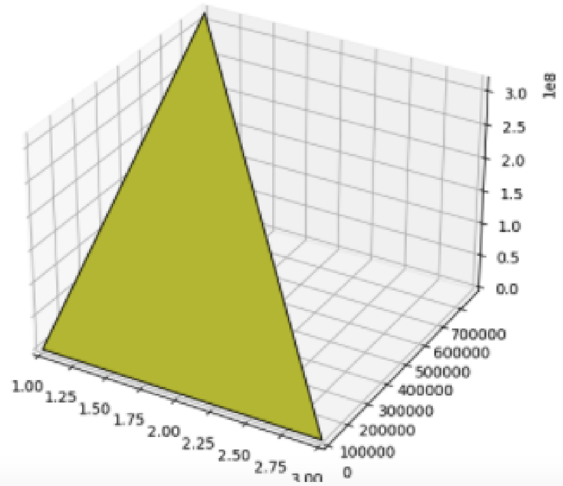
```
3
5
-3 4 4
-3 7 4
-3 8 3
-3 8 1
-3 4 1
3
1 0 0
1 785300 314159265
3 0 0
8
0 0 1
6 6 43
6 8 51
3 11 54
0 8 33
-3 11 36
-6 8 15
-6 6 7
```

```
6
19632
65
```

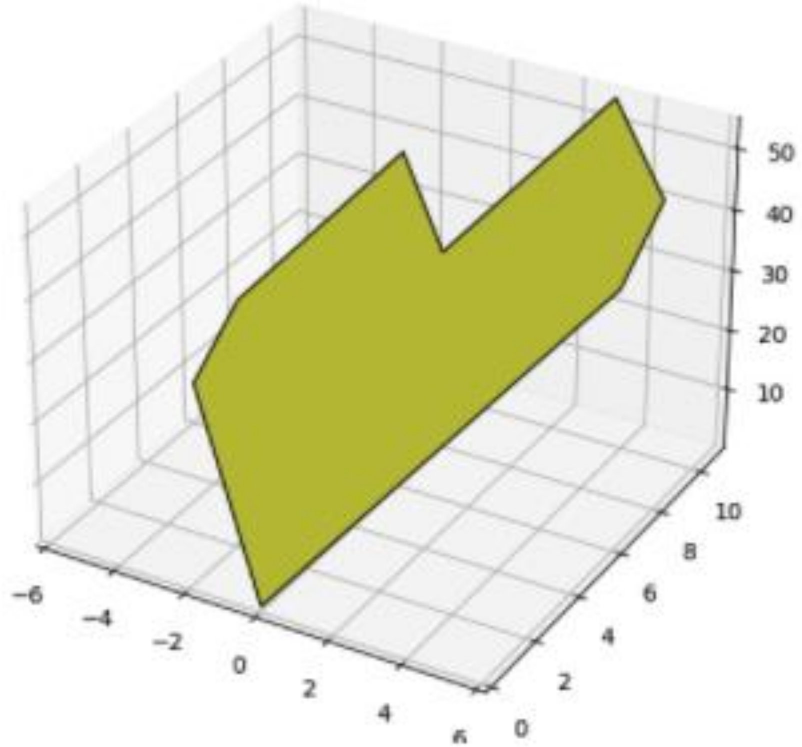
- Case 1: The polygon is a 4×3 rectangle on a plane parallel to YZ plane, but with one trimmed corner. It has 6 (2×3) interior lattice points.



- Case 2: The polygon is a very tall triangle. Only the bottom half contains interior lattice points - one per line.



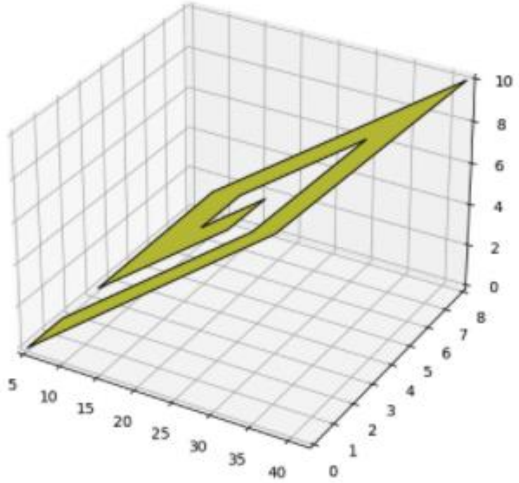
- Case 3: The polygon is a heart shaped non-convex polygon with 65 interior lattice points.



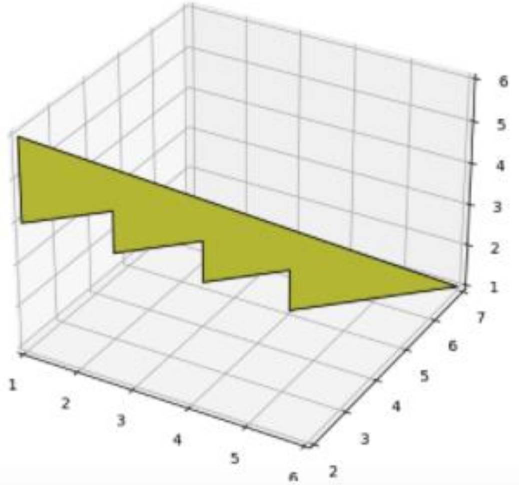
```
3
12
5 0 0
10 0 2
19 6 2
34 6 8
28 2 8
23 2 6
26 4 6
21 4 4
15 0 4
30 0 10
42 8 10
17 8 0
9
1 2 6
1 2 4
2 3 4
2 3 3
3 4 3
3 4 2
4 5 2
4 5 1
6 7 1
5
1 1 2
1 3 4
1 5 6
5 5 10
5 1 6
```

```
14
0
9
```

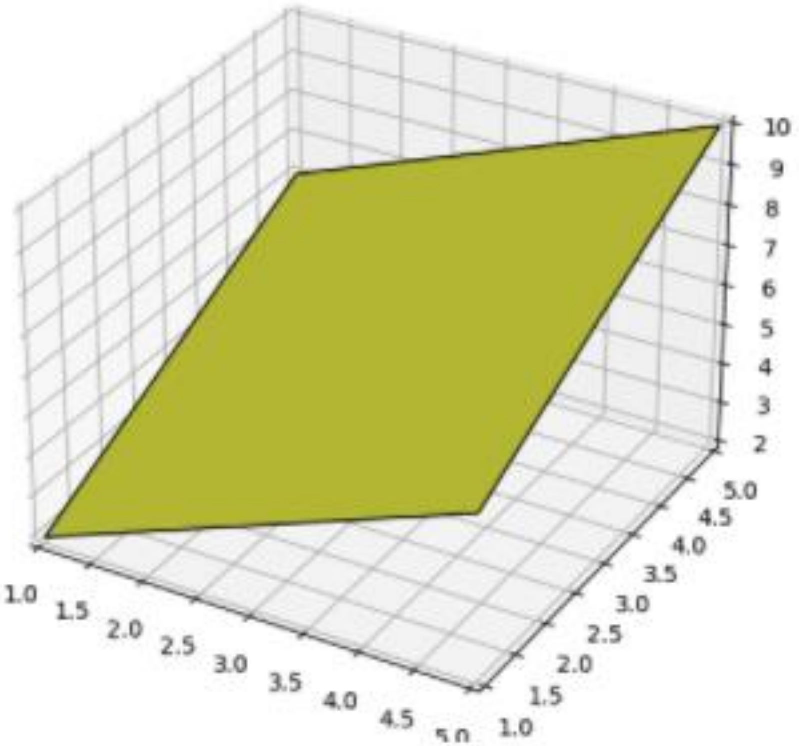
- Case 1: The polygon is a spiral with 14 interior lattice points. Note that there are 6 lattice points within the convex hull of the shape that are outside the polygon.



- Case 2: The polygon is a staircase with no interior lattice points.

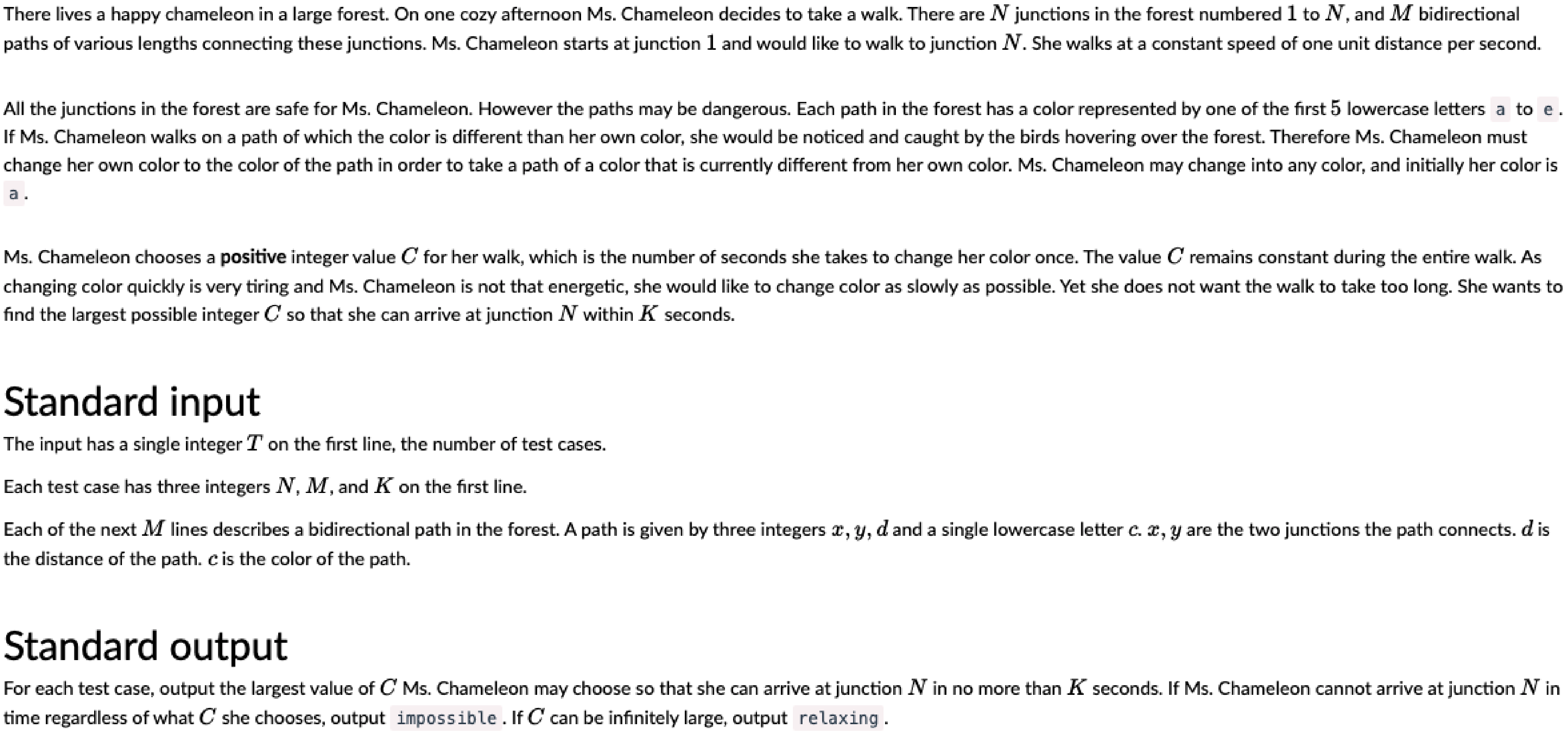


- Case 3: The polygon is a degenerate pentagon - actually a square with 9 interior lattice points.



Chameleon Walk

Time limit: *7280 ms*
Memory limit: *264 MB*



There lives a happy chameleon in a large forest. On one cozy afternoon Ms. Chameleon decides to take a walk. There are N junctions in the forest numbered 1 to N, and M bidirectional paths of various lengths connecting these junctions. Ms. Chameleon starts at junction 1 and would like to walk to junction N. She walks at a constant speed of one unit distance per second.

All the junctions in the forest are safe for Ms. Chameleon. However the paths may be dangerous. Each path in the forest has a color represented by one of the first 5 lowercase letters `a` to `e`. If Ms. Chameleon walks on a path of which the color is different than her own color, she would be noticed and caught by the birds hovering over the forest. Therefore Ms. Chameleon must change her own color to the color of the path in order to take a path of a color that is currently different from her own color. Ms. Chameleon may change into any color, and initially her color is `a`.

Ms. Chameleon chooses a **positive** integer value C for her walk, which is the number of seconds she takes to change her color once. The value C remains constant during the entire walk. As changing color quickly is very tiring and Ms. Chameleon is not that energetic, she would like to change color as slowly as possible. Yet she does not want the walk to take too long. She wants to find the largest possible integer C so that she can arrive at junction N within K seconds.

Standard input

The input has a single integer T on the first line, the number of test cases.

Each test case has three integers N, M, and K on the first line.

Each of the next M lines describes a bidirectional path in the forest. A path is given by three integers x, y, d and a single lowercase letter c. x, y are the two junctions the path connects. d is the distance of the path. c is the color of the path.

Standard output

For each test case, output the largest value of C Ms. Chameleon may choose so that she can arrive at junction N in no more than K seconds. If Ms. Chameleon cannot arrive at junction N in time regardless of what C she chooses, output `impossible`. If C can be infinitely large, output `relaxing`.

Constraints and notes

- $1 \le T \le 10$
- $2 \le N \le 2 \cdot 10^4$
- $1 \le M \le 2 \cdot 10^4$
- $1 \le K \le 10^9$
- For each path, $1 \le x, y \le N, x \ne y, 1 \le d \le 10^4$. The color c is a lowercase letter among `a`, `b`, `c`, `d`, and `e`.
- There can be multiple paths between a same pair of junctions.
- It is guaranteed that it is possible to reach junction N from junction 1 if all paths can be taken regardless of their colors.
- For 66% of the test data, it is guaranteed that if there exists some value of C that works, then the largest feasible value of C does not exceed 50. In other words, if the answer is neither `impossible` nor `relaxing`, then the answer is an integer between 1 and 50.

Input

```
5
5 6 60
1 2 20 a
1 3 40 a
2 3 10 b
2 4 10 b
5 4 10 a
5 3 20 b
5 6 50
1 2 20 a
1 3 40 a
2 3 10 b
2 4 10 b
5 4 10 a
5 3 20 b
2 1 10
1 2 10 b
2 1 10
1 2 20 a
2 1 10
1 2 10 a
```

Output

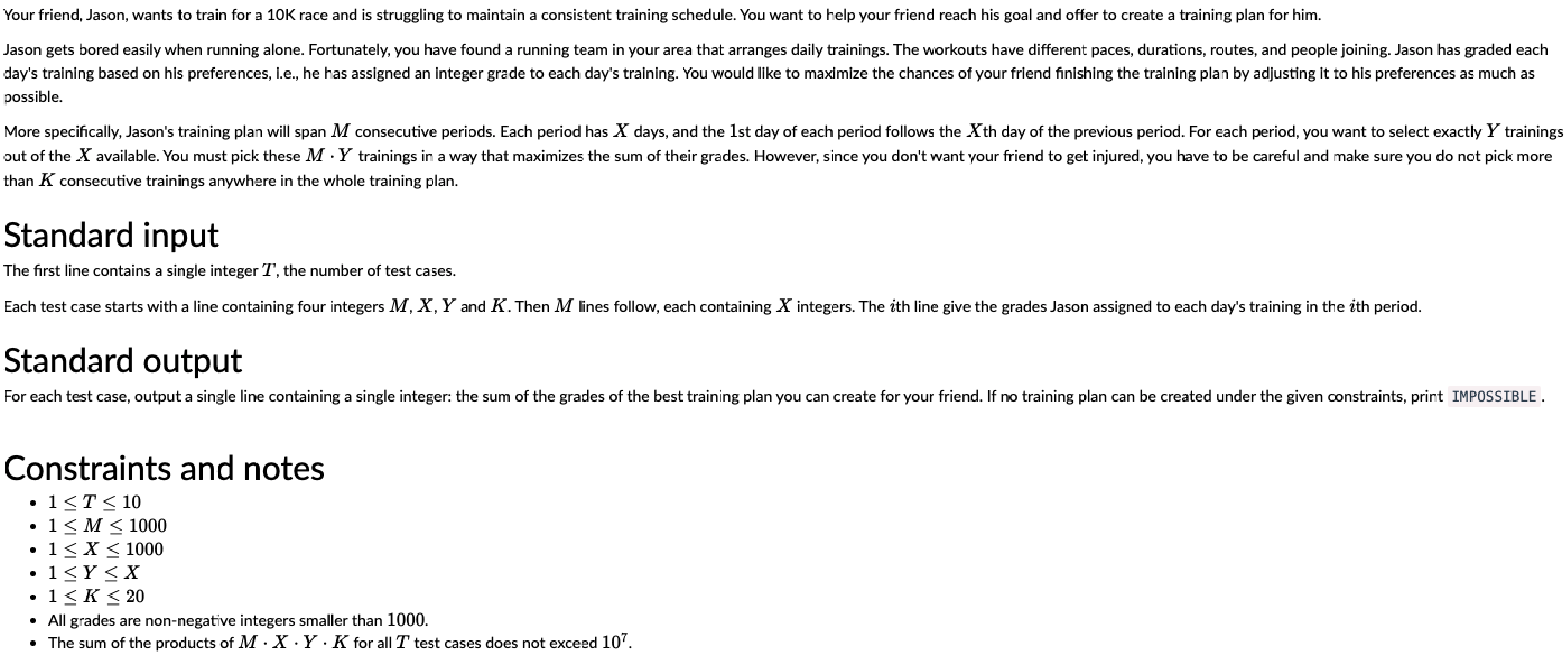
```
10
5
impossible
impossible
relaxing
```

Explanation

- Case 1: Ms. Chameleon can take the path $1 \to 2 \to 3 \to 5$. She needs to change color once using 10 seconds before taking path $(2, 3)$.
- Case 2: This test case has the same forest as case 1. Ms. Chameleon must take the path $1 \to 2 \to 4 \to 5$. She must change color twice along the way and change color more quickly. She cannot take the path in sample 1, because that path is too long and does not allow her to arrive at the destination in time even if she chooses $C = 1$.
- Case 3: Ms. Chameleon must be able to change color instantly ($C = 0$) in order to arrive at junction N in time. However, C has to be a positive integer.
- Case 4: The only path has distance 20, and there is no way to reach junction N in 10 seconds.
- Case 5: Ms. Chameleon does not need to change color and can still arrive at junction N in time. So C can be infinitely large.

Training Plan

Time limit: *4880 ms*
Memory limit: *264 MB*



Your friend, Jason, wants to train for a 10K race and is struggling to maintain a consistent training schedule. You want to help your friend reach his goal and offer to create a training plan for him.

Jason gets bored easily when running alone. Fortunately, you have found a running team in your area that arranges daily trainings. The workouts have different paces, durations, routes, and people joining. Jason has graded each day's training based on his preferences, i.e., he has assigned an integer grade to each day's training. You would like to maximize the chances of your friend finishing the training plan by adjusting it to his preferences as much as possible.

More specifically, Jason's training plan will span M consecutive periods. Each period has X days, and the 1st day of each period follows the Xth day of the previous period. For each period, you want to select exactly Y trainings out of the X available. You must pick these $M \cdot Y$ trainings in a way that maximizes the sum of their grades. However, since you don't want your friend to get injured, you have to be careful and make sure you do not pick more than K consecutive trainings anywhere in the whole training plan.

Standard input

The first line contains a single integer T, the number of test cases.

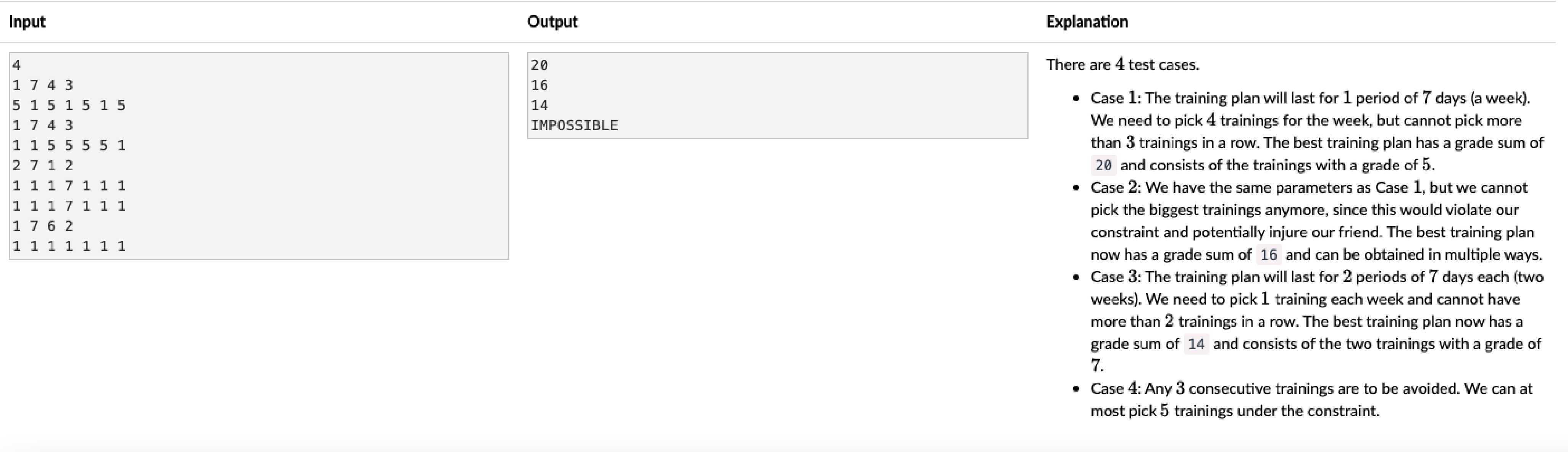
Each test case starts with a line containing four integers M, X, Y and K. Then M lines follow, each containing X integers. The ith line give the grades Jason assigned to each day's training in the ith period.

Standard output

For each test case, output a single line containing a single integer: the sum of the grades of the best training plan you can create for your friend. If no training plan can be created under the given constraints, print `IMPOSSIBLE`.

Constraints and notes

- $1 \leq T \leq 10$
- $1 \leq M \leq 1000$
- $1 \leq X \leq 1000$
- $1 \leq Y \leq X$
- $1 \leq K \leq 20$
- All grades are non-negative integers smaller than 1000.
- The sum of the products of $M \cdot X \cdot Y \cdot K$ for all T test cases does not exceed 10^7.



Input	Output	Explanation

4
1 7 4 3
5 1 5 1 5 1 5
1 7 4 3
1 1 5 5 5 5 1
2 7 1 2
1 1 1 7 1 1 1
1 1 1 7 1 1 1
1 7 6 2
1 1 1 1 1 1 1
	20
	16
	14
	IMPOSSIBLE
		There are 4 test cases.
		- Case 1: The training plan will last for 1 period of 7 days (a week). We need to pick 4 trainings for the week, but cannot pick more than 3 trainings in a row. The best training plan has a grade sum of `20` and consists of the trainings with a grade of 5.
		- Case 2: We have the same parameters as Case 1, but we cannot pick the biggest trainings anymore, since this would violate our constraint and potentially injure our friend. The best training plan now has a grade sum of `16` and can be obtained in multiple ways.
		- Case 3: The training plan will last for 2 periods of 7 days each (two weeks). We need to pick 1 training each week and cannot have more than 2 trainings in a row. The best training plan now has a grade sum of `14` and consists of the two trainings with a grade of 7.
		- Case 4: Any 3 consecutive trainings are to be avoided. We can at most pick 5 trainings under the constraint.

Coloring Tiles

Time limit: *7280 ms*

Memory limit: *264 MB*

There is a wall of N rows and M columns of tiles. Some tiles on the wall are *colorable*, while the others are *non-colorable*. You are to color all the colorable tiles using C types of color pigments. Each type of color pigment has a unique color. Every colorable cell needs to be in one of these C colors. Additionally, there should be no three colorable cells within any 2×2 square that have a same color. In how many ways (modulo $1\,000\,000\,007$) can you color the entire wall?

Standard input

The input has a single integer T on the first line, the number of test cases.

Each test case has three integers N, M, C on the first line. The next N lines each has M characters describing one row of the wall. Each character is either a dot `.`, denoting a colorable tile, or a hash `#`, denoting a non-colorable tile.

Standard output

For each test case, output the number of ways to color the wall modulo $1\,000\,000\,007(10^9+7)$ on a single line.

Constraints and notes

- $1 \leq T \leq 10$
- $N, M \geq 2$
- $N \times M \leq 75$
- $2 \leq C \leq 4$
- There is at least one colorable tile on the wall.
- For 50% of the test files, $N = 2$.

Input

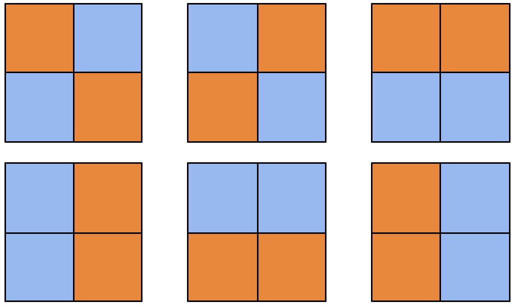
```
3
2 2 2
..
..
2 3 3
...
...
2 6 4
..#..#
...#..
```

Output

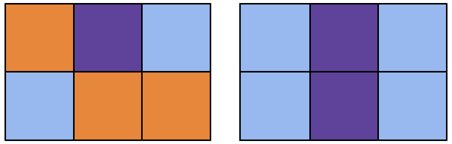
```
6
342
177840
```

Explanation

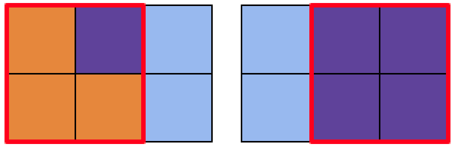
- Case 1: The illustration shows the 6 different ways of coloring a 2×2 wall using $C = 2$ colors.



- Case 2: There are 342 different ways, which are too many to enumerate here. Here are two valid ways to color the 2×3 wall with $C = 3$ colors:



Here are two invalid ways. They are invalid because the highlighted red 2×2 square contains three cells of a same color.



Bridge Construction

Time limit: *6080 ms*
Memory limit: *520 MB*

There are two islands. There are N_1 villages on the first island connected by $N_1 - 1$ bidirectional roads. There are N_2 villages on the second island connected by $N_2 - 1$ bidirectional roads. It is possible to travel between any pair of two villages on a same island via the roads. It takes exactly one unit of time to travel through any single road on either island.

The villagers plan to build a bidirectional bridge between the two islands that connects one village from one island to another village from the other island. It will take exactly one unit of time to travel through the bridge. After the bridge is built, it will be possible to travel between any pair of the $N_1 + N_2$ villages. The villagers would like to build the bridge so that the maximum travel time between any pair of villages is minimized. Where should they build the bridge?

Standard input

The first line of the input has a single integer T, the number of test cases.

Each test case first describes the roads on the first island. It starts with a single integer N_1 on the single line. The villages on the first island are numbered from 1 to N_1. The next following $N_1 - 1$ lines describe the roads on the first island. Each line has two integers a and b, meaning that there is a bidirectional road connecting village a and village b on the first island. Roads on the second island are then described in a same format, starting with N_2 on a single line, followed by $N_2 - 1$ lines each with a pair of villages on the second island.

Standard output

For each test case output two lines. The first line contains the smallest maximum travel time between any pair of villages after the bridge is built. The second line has two space-separated integers x, y ($1 \leq x \leq N_1, 1 \leq y \leq N_2$), indicating that the bridge should be built between village x on the first island and village y on the second island. If there exist multiple ways to build the bridge, you can output any x, y that yields a smallest maximum travel time.

Constraints and notes

- $1 \leq T \leq 10$
- $2 \leq N_1, N_2 \leq 10^5$
- For 60% of the test files, $N_1, N_2 \leq 50$.

Input	Output	Explanation

Input

```
3
8
1 2
2 3
3 4
4 5
2 6
3 7
4 8
4
1 2
1 3
1 4
7
1 2
2 3
3 4
5 6
6 3
3 7
4
1 2
2 3
3 4
2
1 2
2
2 1
```

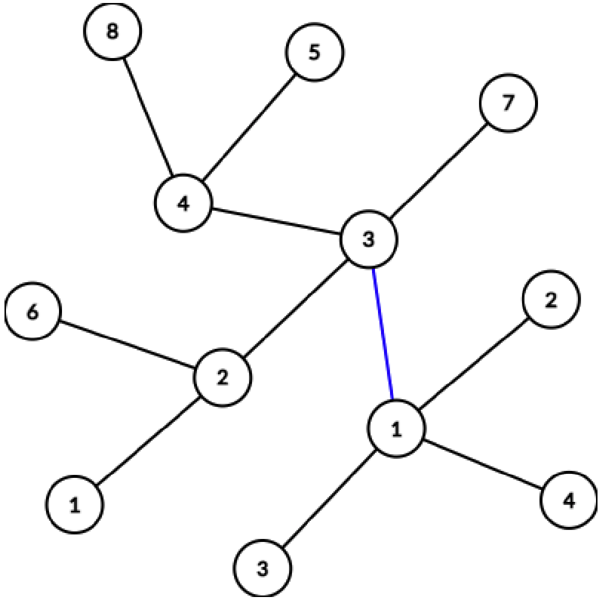
Output

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

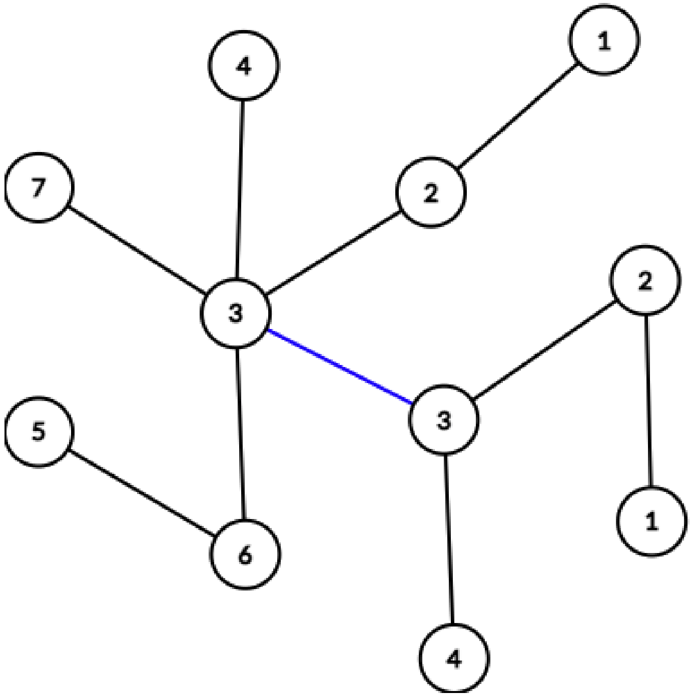
Explanation

There are 3 test cases.

- Case 1: The two islands and their roads are shown below. One optimal way to build the bridge (shown in blue) is to connect village 3 from the first island to village 1 of the second island.



- Case 2: The two islands and the bridge are shown similarly as in Case 1.

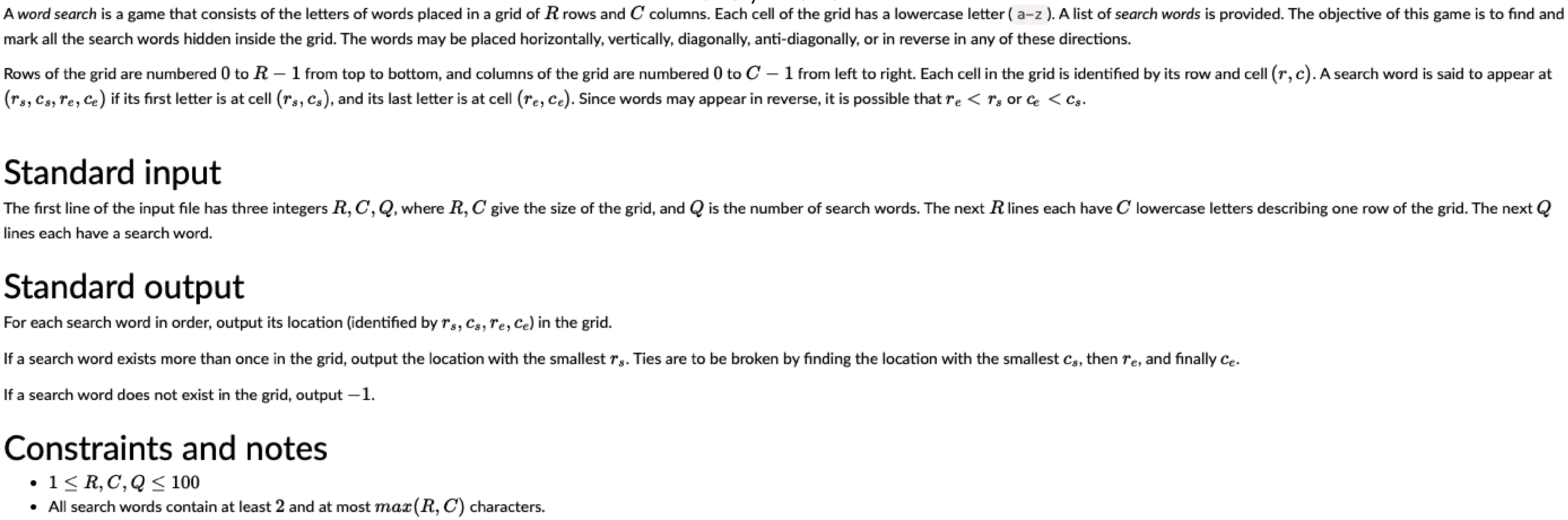


- Case 3: Connecting any pair of villages can yield the smallest maximum travel time of 3, so answers other than `2 2` may also be accepted.

Word Search

Time limit: *4880 ms*

Memory limit: *264 MB*



A *word search* is a game that consists of the letters of words placed in a grid of R rows and C columns. Each cell of the grid has a lowercase letter (`a–z`). A list of *search words* is provided. The objective of this game is to find and mark all the search words hidden inside the grid. The words may be placed horizontally, vertically, diagonally, anti-diagonally, or in reverse in any of these directions.

Rows of the grid are numbered 0 to $R-1$ from top to bottom, and columns of the grid are numbered 0 to $C-1$ from left to right. Each cell in the grid is identified by its row and cell (r,c). A search word is said to appear at (r_s, c_s, r_e, c_e) if its first letter is at cell (r_s, c_s), and its last letter is at cell (r_e, c_e). Since words may appear in reverse, it is possible that $r_e < r_s$ or $c_e < c_s$.

Standard input

The first line of the input file has three integers R, C, Q, where R, C give the size of the grid, and Q is the number of search words. The next R lines each have C lowercase letters describing one row of the grid. The next Q lines each have a search word.

Standard output

For each search word in order, output its location (identified by r_s, c_s, r_e, c_e) in the grid.

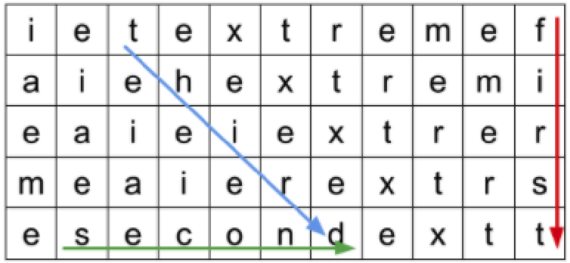
If a search word exists more than once in the grid, output the location with the smallest r_s. Ties are to be broken by finding the location with the smallest c_s, then r_e, and finally c_e.

If a search word does not exist in the grid, output -1.

Constraints and notes

- $1 \leq R, C, Q \leq 100$
- All search words contain at least 2 and at most $max(R, C)$ characters.

Input	Output	Explanation

5 11 4
ietextremef
aiehextremi
eaieiextrer
meaierextrs
esecondextt
first
second
third
fourth
	0 10 4 10
	4 1 4 6
	0 2 4 6
	-1
		There are 4 queries words. The first 3 words can be found in the grid:
		
		i e t e x t r e m e f
		a i e h e x t r e m i
		e a i e i e x t r e r
		m e a i e r e x t r s
		e s e c o n d e x t t
		- The word `first` starts at row 0 column 10, and ends at row 4 column 10 (shown in red).
		- The word `second` starts at row 4 column 1, and ends at row 4 column 6 (shown in green).
		- The word `third` starts at row 0 column 2, and ends at row 4 column 6 (shown in blue).
		- The word `fourth` does not appear in the grid.

4 6 4
xtreme
xtreme
xtreme
iiiiii
xtreme
ieee
eme
ie
	0 0 0 5
	3 3 0 3
	0 3 0 5
	3 2 2 3
		Do not forget to handle reverse order. Make sure that you break the ties correctly.

Image Convolution

Time limit: *9680 ms*
Memory limit: *264 MB*

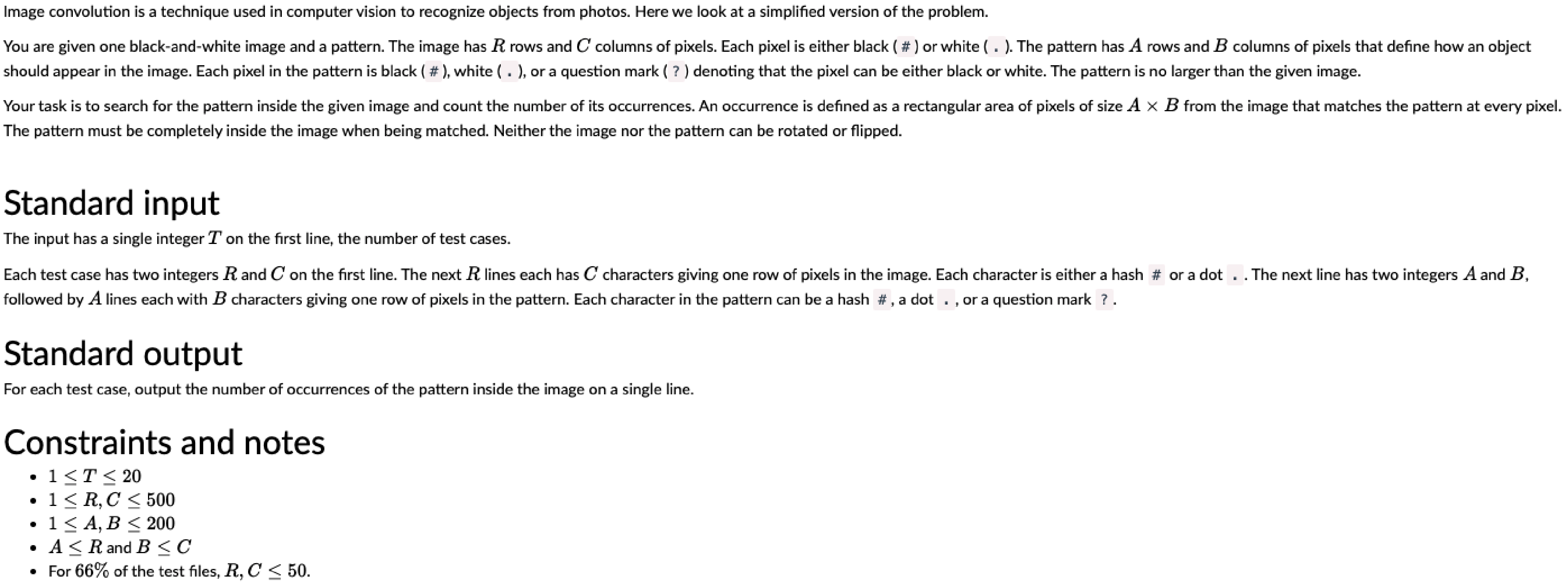


Image convolution is a technique used in computer vision to recognize objects from photos. Here we look at a simplified version of the problem.

You are given one black-and-white image and a pattern. The image has R rows and C columns of pixels. Each pixel is either black (`#`) or white (`.`). The pattern has A rows and B columns of pixels that define how an object should appear in the image. Each pixel in the pattern is black (`#`), white (`.`), or a question mark (`?`) denoting that the pixel can be either black or white. The pattern is no larger than the given image.

Your task is to search for the pattern inside the given image and count the number of its occurrences. An occurrence is defined as a rectangular area of pixels of size $A \times B$ from the image that matches the pattern at every pixel. The pattern must be completely inside the image when being matched. Neither the image nor the pattern can be rotated or flipped.

Standard input

The input has a single integer T on the first line, the number of test cases.

Each test case has two integers R and C on the first line. The next R lines each has C characters giving one row of pixels in the image. Each character is either a hash `#` or a dot `.`. The next line has two integers A and B, followed by A lines each with B characters giving one row of pixels in the pattern. Each character in the pattern can be a hash `#`, a dot `.`, or a question mark `?`.

Standard output

For each test case, output the number of occurrences of the pattern inside the image on a single line.

Constraints and notes

- $1 \leq T \leq 20$
- $1 \leq R, C \leq 500$
- $1 \leq A, B \leq 200$
- $A \leq R$ and $B \leq C$
- For 66% of the test files, $R, C \leq 50$.

Input

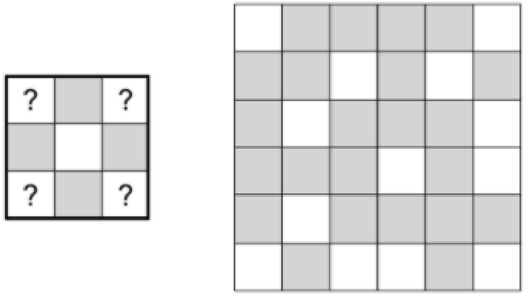
```
3
6 6
.####.
##.#.#
#.###.
###.#.
#.####
.#..#.
3 3
?#?
#.#
?#?
4 4
#.#.
.###
###.
.#.#
2 2
.#
##
1 4
#.#.
1 2
??
```

Output

```
5
2
3
```

Explanation

There are 3 test cases. Here we explain the first test case. The pat
and image are as follows:



The five occurrences of the pattern within the image are: