



IEEE Xtreme 14.0 Problems [24 October 2020]

Contents

Welcome to IEEE Xtreme 14.0	3
IEEE Xplore Indexing.....	5
Linearly Separable Samples.....	9
Mosaic Decoration I	11
Identifying Infected	11
Hotel Wiring	15
Making a Tangram.....	17
Crafting Wooden Tables.....	19

The Defensive Walls	21
Rotational Lights.....	23
Magical Stones I.....	26
Parkour.....	28
Non-Overlapping Palindromes.....	32
Furin Back	33
Molecules.....	36
Mosaic Decoration II	38
The Last of Us.....	41
ARM Constant Multiplication.....	45
Game of Life 2020.....	48
Social Distancing in Class	50
Poker Game.....	52
Magical Stones II.....	54
Rescue Mission.....	56
Restaurant Reopening.....	58
Coin Collector.....	60
Mosaic Decoration III	62
Coupon Codes	64

Welcome to IEEEExtreme 14.0

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

On behalf of the executive committee, Welcome to the 14th IEEEExtreme Programming contest! Our slogan for IEEEExtreme 14.0 is **You Are Not Alone** and with over 10\,00010000 participants and volunteers, this has never been truer. It is important that you enjoy this event as much as you can. Hundreds of volunteers have been working for months to prepare this programming party for you. Be sure to get plenty of rest, stay hydrated, and don't forget to eat. Above all, have as much fun as possible!

During the next 2424 hours, challenges will be released at regular intervals. The first 33 challenges will show up at 55 minutes into the contest. Then starting at 22 hours into the contest, we will release at least one problem at each round o'clock, that is, 33 hours, 44 hours, and so on, up to 2323 hours. 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. In the event that 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. 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 try to solve every problem, even if your solution is not optimal or even complete. You will be able to obtain a partial score by solving a subset of test files on any given challenge.

Finally, a special thank you to all the volunteers who have made this Xtreme 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.

Your first task is to write a program that outputs some text with your feedback and submit it to the system. This task will not be scored and it is optional. There will be 55 minutes before the real tasks appear. Please do take the time to write a message, and we look forward to hearing from you!

Good luck and have fun!

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.

Notes

- 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.

IEEE Xplore Indexing

Time limit: 1280 ms
Memory limit: 264 MB

Warm greetings to all IEEEExtreme Participants from the Xplore API Team!

As part of the IEEEExtreme competition, in this challenge you are asked to identify the most important index terms of an academic journal. This challenge emulates a real-world application - The IEEE Xplore API uses a similar approach!

For a full dynamic database search IEEE Xplore API is available for your IEEE research needs. Xplore API provides metadata on 4.9M academic works and is now delivering full-text content on 50K *Open Access* articles. Xplore API will meet your research needs fast and easy. The Xplore API Portal supports PHP, Python and Java as well as providing output in Json and XML formats. Many API use cases are listed within the API Portal.

Xplore API registration is free. To learn more about IEEE Xplore API please visit developer.ieee.org and register for an API key TODAY!

Challenge

An index term, subject term, subject heading, or descriptor, in information retrieval, is a term that captures the essence of the topic of a document. Index terms make up a controlled vocabulary for use in bibliographic records. They are an integral part of bibliographic control, which is the function by which libraries collect, organize and disseminate documents.

In this challenge you are given a list of index terms, a list of stop words, and a document. You are asked to parse the document and identify the top 3 index terms that have the highest keyword density.

Definitions

For the purpose of this challenge:

- A *word* is a consecutive sequence of lowercase letters (a-z) or apostrophe ('). A word must not contain any other characters. The sequence must have at least 4 characters.
- A *punctuation* is a comma (,), a period (.), a question mark (?), or an exclamation mark (!).
- A *token* is a consecutive sequence of lowercase or uppercase letters (a-zA-Z), apostrophes ('), or punctuation. A token must not contain spaces or newlines.
- A *index term* is a word. A *stop word* is a word.
- A *document* is a simplified XML with matched tags such as `<body>`, and `</abstract>`. An opening tag has the format `<[a-z]+>`. A closing tag has the format `</[a-z]+>`. In other words, a tag only has lowercase letters in them and is always enclosed by a pair of brackets `<>`.
- There are three *special tags*: `<title>`, `<abstract>`, and `<body>`.
- Tags can be nested, such as `<a><p></p>`.
- A document may have an arbitrary number of tokens, spaces, or newlines between its tags. No tokens appear outside all the tags.

Keyword Density

Here is the methodology to compute the keyword density for an index term w :

- Each token in the document is normalized to its underlying word, by first converting all uppercase letters to lowercase, and then removing all punctuation in it.
- All words that are *stop words* are ignored.
- Compute the total number of words in the special tags of the document as L . A same word may appear multiple times in a special tag and is counted multiple times in L . Note that words outside special tags are not counted towards L .
- Compute the index term score S_w by its occurrences in the special tags: Each occurrence in the `<title>` scores 5. Each occurrence in the `<abstract>` scores 3. Each occurrence in the `<body>` scores 1.
- The *keyword density* of an index term w is defined as $S_w/L \cdot 100$

Standard input

The first line of the input has a list of stop words separated by a single semicolon.

The second line of the input has a list of index terms separated by a single semicolon.

From line three to the end of the input file gives the XML document.

Standard output

Output the top 3 index terms with the highest keyword density. Output each index term along with its keyword densities on a single line, with a colon and a space between them (see the sample). The index terms should be sorted by decreasing keyword density. Your keyword density is considered correct if it has an absolute or relative error of at most 10^{-6} from the correct keyword density.

Output only the index terms with a positive keyword density. If there are fewer than 3 index terms that appear in the document, output only those that appear.

In case there is a tie on keyword densities, all index terms that have a keyword density that is as large as the 3rd highest keyword density should be printed. If two index terms have a same keyword density, the *lexicographically smaller* word should be printed first.

Constraints and notes

- The number of index terms is between 5 and 30.
- The number of stop words is between 5 and 150.
- All index terms are distinct. All stop words are distinct. No index term is a stop word, and vice versa.
- The document contains exactly one `<title>`, one `<abstract>`, and one `<body>` tag. No special tag is in another special tag. However, a special tag can be nested in other tags.
- The document contains only lowercase letters (`a-z`), uppercase letters (`A-Z`), apostrophes (`'`), punctuation (`, . ? !`), XML tags, spaces (not tabs), or newlines.
- The document contains at most 20 000 characters.
- No line of the document has trailing spaces. However, a line may have leading spaces as indentation.
- No line of the document contains only the newline character. There are no empty lines.
- Each XML tag starts and ends on a same line.
- The input files of this challenge are not only chosen from real-world scenarios, but may also be specially constructed to test that your computation is correct, just as in the other challenges.

Input	Output	Explanation
<pre>being;does;have;haven't;more;should;shouldn't;t classification;cryptography;diseases;probabilit <response> <article> <title>A Novel Approach to Image Classificati <publicationtitle>IEEE Transactions on Cloud <abstract>Classification of items within PDF </article> <body> <sec> <label>I.</label> <p>Should Haven't That is a bunch of text pa <p>I bet diseases you can't find probability <p> <fig> <label>FIGURE.</label> <caption>This is a figure representing con </fig> </p> </sec> </body> </response></pre>	<pre>classification: 19.512195122 stability: 9.756097561 probability: 3.658536585</pre>	<p>There are a total of 82 words. Their scores S_w are:</p> <ul style="list-style-type: none">• classification: 16• stability: 8• probability: 3• cryptography: 1• diseases: 1 <p>For <code>classification</code>, the keyword density is $16/82 \cdot 100 \approx 19.512195$. The other keyword densities can be computed similarly.</p>

```
what;when;where;like;that
welcome;ieee;xtreme;ieeextreme;programming
<title>Welcome to IEEEXtreme!</title>
<keyword>welcome, ieeextreme</keyword>
<abstract>
Welcome!Participants!!!
IEEE Xtreme is a global challenge in which team
Compete in a twentyfour hour timespan against e
</abstract>
<body>WELCOME. wel.come... Are you ready? Good
<other>
Mark your calender and don't miss the action.
</other>
```

```
ieee: 30.000000000
ieeextreme: 20.000000000
welcome: 17.500000000
xtreme: 17.500000000
```

Make sure that you read the problem statement correctly. Do not forget to handle the tied index terms and output them in lexicographical order.

Linearly Separable Samples

Time limit: 6080 ms

Memory limit: 264 MB

A linear classifier, especially the Support Vector Machine (SVM), is a popular tool in the task of pattern recognition and artificial intelligence. An essential problem for the linear classifier is whether the given samples are linearly separable.

Let's consider the 2D cases without the bias term. Each sample has two features, $x_1 \in \mathbb{R}$ and $x_2 \in \mathbb{R}$, and a label $y \in \{1, -1\}$. You will be given T queries. For each query you are given a group of points on a 2D plane, your task is to determine whether the positive samples (labelled 1) and the negative samples (labelled -1) can be separated by a line through the origin $(0, 0)$. If a line goes through some positive and/or negative points but separates all other points, it can be also accepted.

Mathematically speaking, you are asked to find whether there exist $w_1 \in \mathbb{R}$ and $w_2 \in \mathbb{R}$, s.t.

$$\langle [w_1, w_2], [x_1, x_2] \rangle = w_1 x_1 + w_2 x_2 \geq 0 \text{ IFF } y = 1$$

$$\langle [w_1, w_2], [x_1, x_2] \rangle = w_1 x_1 + w_2 x_2 \leq 0 \text{ IFF } y = -1$$

hold for all samples $(x_1, x_2), y$.

Standard input

The first line of the input has an integer T which represents the number of queries. Then, T queries follow. The first line of each query has an integer N that denotes the number of points in the query. Each of the following N rows has two floating point numbers and an integer separated by single spaces. They represent the two features x_1, x_2 and the label y of one point.

Standard output

For each query, output the answer on a single line. Output **YES** if the given points can be separated by a line through the origin, and **NO** otherwise.

Constraints and notes

- $1 < T \leq 10$
- $1 \leq N \leq 10^5$
- $-100.0 \leq x_1, x_2 \leq 100.0$
- $y = 1$ or $y = -1$
- x_1, x_2 are given with at most two decimal points.
- Points may overlap and have the same x_1, x_2 .
- For 50% of the test data, $1 \leq N \leq 5\,000$

Input

```

2
9
-2.0 1.0 1
2.0 2.0 -1
4.0 2.5 -1
4.0 0.5 -1
-1.0 -1.0 1
4.0 1.5 -1
-3.0 -1.0 1
2.0 3.0 -1
-4.0 1.0 1
4
2.0 2.0 1
2.0 -2.0 -1
-2.0 2.0 -1
-2.0 -2.0 1

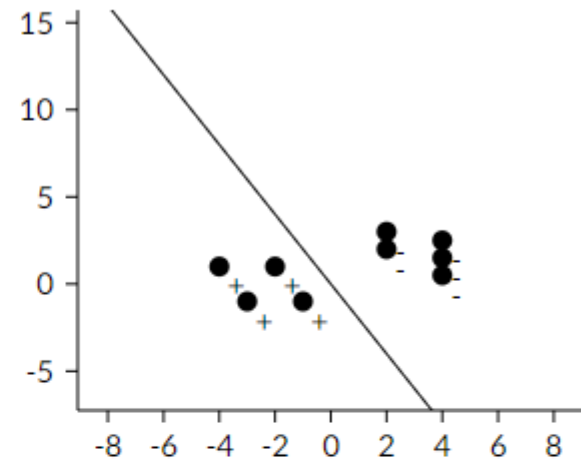
```

Output

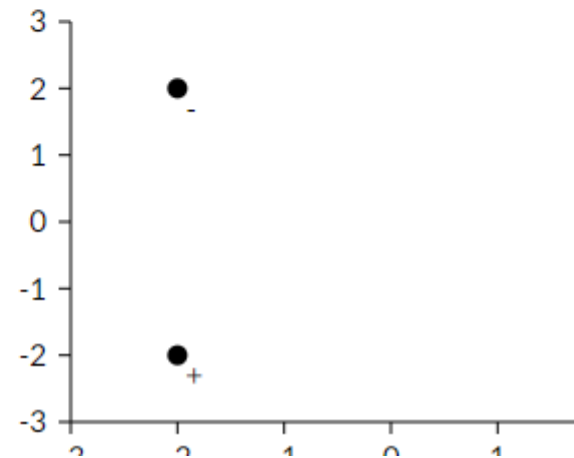
```

YES
NO

```

Explanation

For the first query, it can be seen from the figure that those points can be separated by the line there.



Mosaic Decoration I

Time limit: 1280 ms
Memory limit: 264 MB

Zapray lives in a big mansion that has N bathrooms. He wants to decorate the bathroom walls using mosaic tiles of two colors: black and pink. The i th bathroom needs B_i black tiles and P_i pink tiles. Mosaic tiles are sold in piles. Zapray can buy one pile of 10 black tiles for C_B dollars, and one pile of 10 pink tiles for C_P dollars. How much money does he need in total to decorate all the N bathrooms?

Standard input

The input contains three integers N, C_B, C_P on the first line.

The next N lines each have two integers. The i th line has B_i and P_i .

Standard output

Output a single integer, the amount of money in dollars that Zapray needs to decorate all his bathrooms.

Constraints and notes

- $2 \leq N \leq 100$
- $1 \leq C_B, C_P \leq 1000$
- $1 \leq B_i, P_i \leq 1000$

Input	Output	Explanation
<pre>3 5 7 10 10 20 30 30 3</pre>	<pre>65</pre>	There are 3 bathrooms to decorate. In total 60 black tiles and 43 pink tiles are needed. Zapray needs to purchase 6 piles of black tiles and 5 piles of pink tiles. The total cost is $6 \times 5 + 5 \times 7 = 65$ dollars.

Identifying Infected

Time limit: 2480 ms
Memory limit: 264 MB

Bad-virus is a highly contagious virus that can seriously complicate people's health. This virus is only spread from person to person and the only way to prevent infection (not 100% safe) is by wearing masks.

Chief of epidemiology has received the first case of bad-virus in your country, this case is imported from the other country. In this context, in coordination with the president, it was decided to close the country's borders, prohibit encounters of more than two persons and make the use of masks mandatory. But in order not to affect the economy, no type of quarantine will be carried out.

Two days later, Chief of epidemiology has been notified about the second case of bad-virus (now necessarily is a national case). Knowing that there are N persons in the country (including the imported case) enumerated by an integer from 1 to N , they ask you to identify all people who necessarily are infected. For this task, the department of epidemiology provides you with a list of the M encounters of two persons during this time. You will be given Q scenarios that you should handle separately. In the i -th scenario, you will be given the IDs of the first and the second case of bad-virus, and you should count how many people that can be 100% identified as infected (including the first and the second case).

Standard input

The first line contains two integers N and M , denoting the number of persons and number of encounters respectively.

Each one of the next M lines contains two different integers A_i and B_i , denoting the two persons in the encounter (there aren't two equal pairs of integers).

The next line contains an integer Q , denoting the number of scenarios.


Each one of the next Q lines contains two different integers F_i and S_i , denoting the first and second case of bad-virus in the i -th scenario.

Standard output

For each scenario, output the number of people that can be 100% identified as infected on a single line.

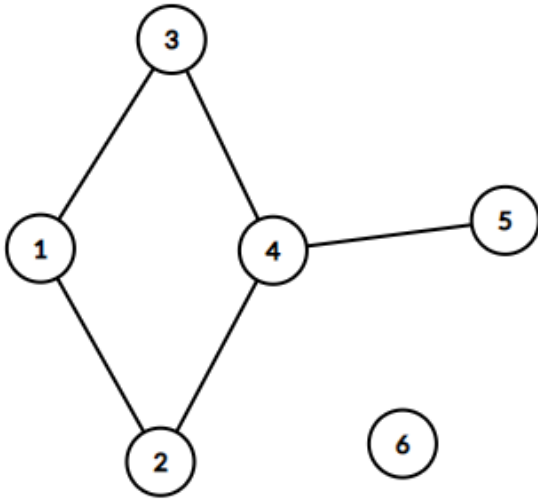
Constraints and notes

- $2 \leq N \leq 10^5$
 - $1 \leq M, Q \leq 2 \times 10^5$
 - $1 \leq A_i < B_i \leq N$ for all $1 \leq i \leq M$
 - $A_i \neq A_j$ or $B_i \neq B_j$ for all $1 \leq i < j \leq M$
 - $1 \leq F_i, S_i \leq N$ and $F_i \neq S_i$ for all $1 \leq i \leq Q$
 - For each scenario, it is guaranteed that there is at least a sequence of encounters so that the second case is originally infected from the first case.
-
- For 30% of the test data, $N \leq 100$ and $M, Q \leq 200$.
 - For 60% of the test data, $N \leq 1\,000$ and $M, Q \leq 2\,000$.

Input	Output	Explanation
6 5 1 2 1 3 2 4 3 4 4 5 2 1 5 4 3	3 2	Below is the illustration of the encounters. 

Explanation

Below is the illustration of the encounters.



In the first scenario, there are only two ways such that the fifth person can be infected by the first person, i.e. $1 \rightarrow 2 \rightarrow 4 \rightarrow 5$ and $1 \rightarrow 3 \rightarrow 4 \rightarrow 5$. We cannot identify which one of the second and the third person that is actually infected. So, the people that can be 100% identified as infected are 1, 4, and 5.

In the second scenario, the people that can be 100% identified as infected are 3 and 4.

Hotel Wiring

Time limit: 2480 ms

Memory limit: 264 MB

A construction company has decided to let an inexperienced electrician work on power supply wiring for their newest hotel building. After the construction and wiring of the building has been completed, it was identified that the electrician has made a mistake in wiring and it will take a significant amount of time to fix the issue.

The hotel consists of M floors and each floor has N rooms. Power supply to rooms is controlled by one master switch on each floor. By design, power is supplied to all the rooms on a floor if and only if the master switch of that floor is on. However due to the electrician's mistake in wiring, some rooms have incorrect wiring and their power is supplied the other way around. A room with incorrect wiring receives power if and only if the master switch of its floor is off.

There can be both correctly and incorrectly wired rooms on a same floor. For example, consider a floor has 4 rooms with correct wiring and 2 rooms with incorrect wiring. If the master switch of that floor is on, 4 rooms will have power and 2 rooms will not have power. If the master switch is off, then, 4 rooms will not have power and 2 rooms will have power.

The company does not want to spend extra money to fix the wiring mistake. Instead, they decide to first turn on all the master switches, and then turn off exactly K of them to maximize the total number of rooms across all floors that will receive power.

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 M , N and K where M is the number of floors in the hotel, N is the number of rooms per floor, and K is the exact number of master switches that will be turned off.

For each of the next M lines, there is a single integer between 0 and N that indicates the number of rooms that are wired correctly on one floor.

Standard output

For each test case, output in a single line the maximum number of rooms that may receive power.

For each test case, output in a single line the maximum number of rooms that may receive power.

Constraints and notes

- $1 \leq T \leq 20$
- $1 \leq K \leq M \leq 10^6$
- $1 \leq N \leq 10^6$
- The sum of M across all test cases in a single test file is at most $2 \cdot 10^6$.

Input	Output	Explanation
<pre>2 2 2 1 2 0 2 4 2 0 3</pre>	<pre>4 5</pre>	<p>There are two test cases in this sample test file.</p> <p>In the first test case, there are two floors with two rooms on each floor. On the first floor, both rooms have correct wiring and on the second floor, both rooms have incorrect wiring. Since $K = 1$, the master switch on the second floor can be turned off to supply power to both rooms. Hence, all rooms in the hotel can receive power.</p> <p>In the second test case, there are two floors with four rooms on each floor. On the first floor, all rooms have incorrect wiring and on the second floor, 1 room has incorrect wiring and 3 have correct wiring. Since $K = 2$, the master switches on both of the floors must be turned off. As a result, the number of rooms with power supply are 4 on the first floor and 1 on the second floor.</p>

Making a Tangram

Time limit: 1280 ms

Memory limit: 264 MB

In this challenge you are to create a beautiful tangram from a board of $N \times N$ cells. The board will be cut into N tangram pieces. Each piece consists of exactly N cells that are 4-connected (up, down, left, right). All the tangram pieces shall together fit into the $N \times N$ board without extra or empty cells. To make the tangram fun, each piece will have a distinct color, and no two pieces shall be of a same shape.

Two tangram pieces are said to have a same shape if one can be rotated clockwise or counterclockwise to look exactly the same as the other. For example, the following four pieces are the same:

```
1  X  X  XX  X
2  XX  XXX  XX  XXX
3  XX  X  X  X
```

These two pieces have different shapes:

```
1  X  X
2  XXX  XXX
```

A tangram piece is allowed to have holes, such as:

```
1  XXX
2  X  X
3  XXX
```

Now it is up to you to design the tangram in whatever way you like!

Standard input

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

Each of the next T lines has one test case with a single integer N , the size of the board.

Standard output

For each test case, output any tangram design that satisfies the requirement. The output has N lines, each with N characters. Mark each tangram piece with any unique character from the alphabet: lowercase letters `a-z`, uppercase letters `A-Z`, or digits `0-9`. Any valid tangram will be accepted.

If there is no way to cut the board to make the tangram, output `impossible` on a single line.

Constraints and notes

- $1 \leq T \leq 20$
- $2 \leq N \leq 62$

Input	Output	Explanation
2 5 3	XXXTr XTTTr XTErr mmEEr mmmEE impossible	<p>In the first test case $N = 5$, note that you may pick any characters from the alphabet to color the tangram pieces.</p> <p>In the second test case $N = 3$, only these two types pieces can be made. Therefore you cannot cut the board into three different pieces.</p> <pre>1 XX 2 X XXX</pre>

Crafting Wooden Tables

Time limit: 1280 ms
Memory limit: 264 MB

A lot of people are stuck at home due to the unfortunate pandemic this year. They thus resort to building their wonderful home on a deserted island. Rolan is such an deserted islander.

To live a fruitful life on a deserted island, Rolan needs to craft a lot of furniture. One of Rolan's favorite, is the *wooden table*. Crafting a wooden table requires C pieces of wood, which is a resource obtainable by chopping trees on the island. Rolan has a pocket of limited size that can hold the wood and the crafted wooden tables. Her pocket has N slots, and each slot can hold either a single wooden table, or a pile of wood with at most P pieces of wood.

Rolan has W pieces of wood at the beginning and can distribute them into the N pocket slots arbitrarily. To craft a wooden table, Rolan takes C pieces of wood from any of the pocket slots, removing them from those slots. If any pile of wood becomes empty, the pocket slot becomes free. If there is no free slot in Rolan's pocket, the crafting stops. Otherwise, the crafted wooden table will be placed in a free pocket slot. Rolan repeats the crafting until no more wooden tables can be crafted.

How many wooden tables will Rolan eventually have?

Standard input

The input has four integers C, N, P, W on a single line.

Standard output

Output a single integer, the number of wooden tables that Rolan can craft.

Constraints and notes

- $1 \leq C, N, P, W \leq 10^{15}$
- $W \leq N \cdot P$
- For 60% of the test data, $C, N, P, W \leq 10^6$

Input	Output	Explanation
4 3 3 8	2	Rolan distributes the wood into the pocket as $[3, 3, 2]$. Each integer denotes the number of wood pieces in a pile. Let T denote a wooden table. The first crafted table will be placed as $[3, 1, T]$. Then a second table can be crafted and placed as $[T, 0, T]$.
2 3 3 8	1	Initial pocket is $[3, 3, 2]$. Crafting a wooden table consumes 2 pieces of wood. Only one table can be crafted and the pocket is $[3, 3, T]$.
10 3 3 8	0	Crafting a wooden table requires 10 pieces of wood. But Rolan does not have that many pieces of wood.

The Defensive Walls

Time limit: 9680 ms

Memory limit: 264 MB

In ancient times, there was the Kingdom of Xtreme whose people lived peacefully side by side with a giant race. But at one time, the peace was destroyed because of the inevitable conflict between humans and giants. Humans are very disadvantaged when fighting directly with the giants, so the king could only decide to evacuate his people to a safe place until the peace is reestablished with the giants.

Xtreme Kingdom had a fortress from the past when there was a first war with the giants. Unfortunately, some parts of the fortress were damaged by time so that the people could only use the parts that were still standing strong, because they did not have time to rebuild it again.

You as the king's advisor are asked to calculate the total area of the fortress that can still protect the people from giant attacks. You are given a 2-dimensional map representing fortress' condition. There are N straight line segments representing the walls of the fortress which still stand firm. The walls are built so big and strong that they are guaranteed not to be climbed or destroyed by giants. Each line segment on the map is either vertical or horizontal, i.e. if the endpoints of i -th line segment are (X_{A_i}, Y_{A_i}) and (X_{B_i}, Y_{B_i}) then either $X_{A_i} = X_{B_i}$ or $Y_{A_i} = Y_{B_i}$.

Standard input

The first line of the input contains one integer N representing the number of walls.

The next N lines describe the existing walls. Each line contains four integers separated by single spaces, X_{A_i} , Y_{A_i} , X_{B_i} , and Y_{B_i} indicating that the i -th wall spans between point (X_{A_i}, Y_{A_i}) and point (X_{B_i}, Y_{B_i}) .

Standard output

Output one integer representing total area of the fortress which can protect the people from the giant attacks. An area is safe from giant attacks if it cannot be reached by the giants from outside the fortress.

Constraints and notes

- $1 \leq N \leq 10^5$
- $0 \leq X_{A_i}, Y_{A_i}, X_{B_i}, Y_{B_i} \leq 10^9$
- Either $X_{A_i} = X_{B_i}$ or $Y_{A_i} = Y_{B_i}$
- $X_{A_i} + Y_{A_i} < X_{B_i} + Y_{B_i}$
- The given input is guaranteed that there are no intersections between every parallel line segments.

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

Input

```

16
1 1 10 1
3 0 3 10
2 10 8 10
8 2 8 10
8 2 12 2
11 2 11 5
4 5 11 5
5 5 5 8
5 8 7 8
7 4 7 9
5 4 7 4
6 2 6 7
9 3 10 3
9 4 10 4
10 3 10 4
9 3 9 4

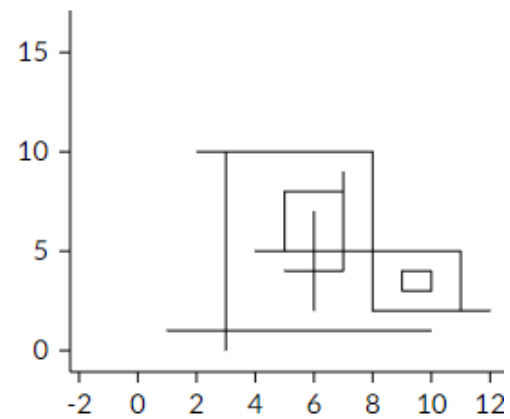
```

Output

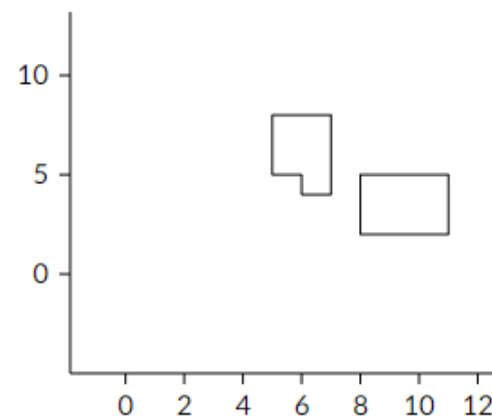
```
16
```

Explanation

Below is the map of the fortress in 2-dimensional coordinates.



The overall protected area can be illustrated as two polygons below.



So, the total area is $7 + 9 = 16$

Rotational Lights

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

Your little brother Jerry is four years old. He's quite fun to be around, unless you have work to do. When this happens, you need something to keep him occupied.

Luckily for you, on his birthday, Jerry got a new game that he can't resist playing. It's called "all the colors of the rainbow" and consists of a circle with T LED lights, evenly distributed on its circumference. Each of the lights has a different color (hence the name) and comes with a switch, so you can either turn it on or off. Four-year old boys get easily excited with these things.

You really don't know the rules of the game but, every time Jerry wants to play, he asks you to provide an initial configuration: turn some of the lights on and leave the rest off. After you do this, Jerry plays the game on his own: he keeps switching lights on and off until he's satisfied with the result and declares victory.

Although you really don't understand your brother's game, you've noticed three important things:

- Some initial configurations are easier than others for Jerry to solve. So, once in a while you pick a hard configuration that keeps Jerry busy for a longer time.
- If a configuration is hard, so are all of its rotations. A rotation of a configuration is what you get by rotating the lights that are turned on by some angle that is a multiple of $2\pi/T$.
- Jerry remembers the configurations he's already played, so if you find a hard one, you cannot directly reuse it. He somehow remembers the combinations of the colors. However, you can still trick your brother by using a rotation, as long as the result is a configuration that he hasn't previously seen.

You just found a new hard configuration that kept Jerry busy for one hour straight! Assuming that Jerry had not played any of this configuration's rotations before, how many such rotations can you use to keep Jerry busy?

Standard input

The first line will contain two space-separated positive integer numbers, N and T : the number of lights that are turned on in the initial configuration and the total number of lights.

The second line will contain N space-separated integer numbers, the positions of the lights that are turned on. These numbers will be in the range $[0, T)$ and will be provided in strictly ascending order. (In our examples, we make the arbitrary assumptions that position 0 is in the north and that positions increase clockwise.)

Standard output

Standard output

Your program must print a single line containing a single integer number: how many different rotations of the initial configuration exist.

Constraints

- $1 \leq N \leq T$.
- For 10% of the test data, $T \leq 10^3$.
- For 40% of the test data, $T \leq 10^6$.
- For 70% of the test data, $N \leq 10^6$ and $T \leq 10^9$.
- For 100% of the test data, $N \leq 10^6$ and $T \leq 10^{18}$.

Figure of Sample Test 0

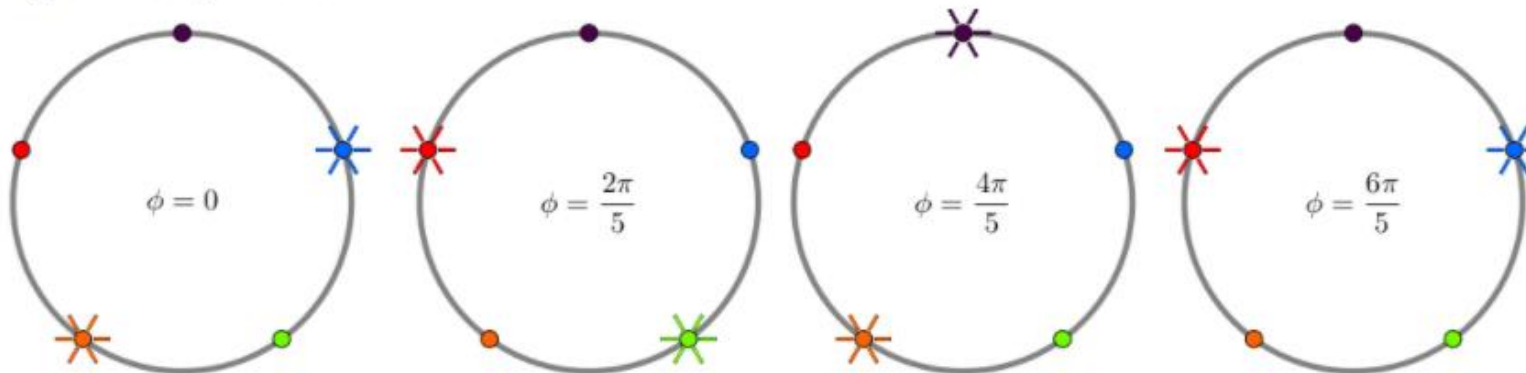


Figure of Sample Test 1

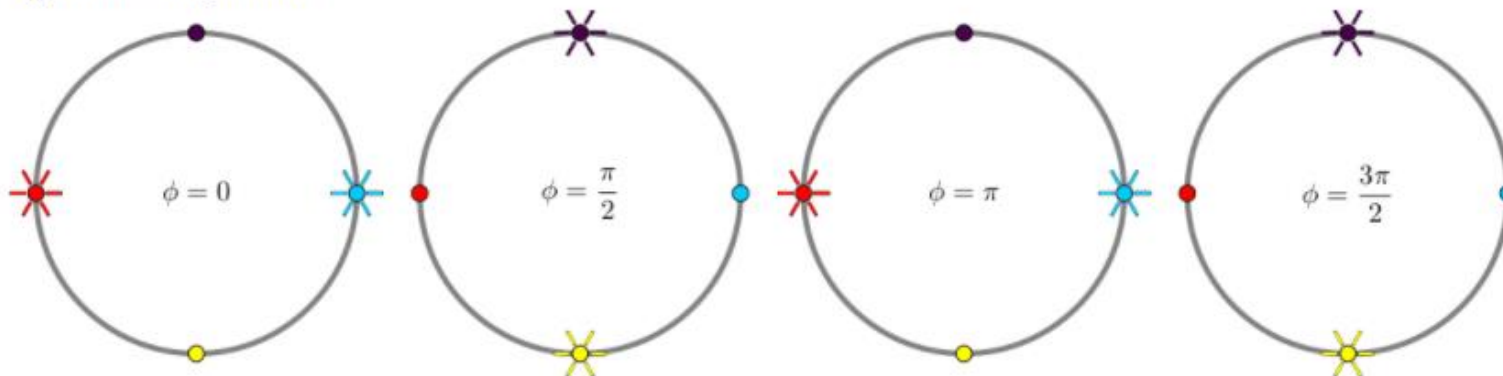


Figure of Sample Test 2



Input	Output	Explanation
<pre>2 5 1 3</pre>	4	<p>In this example, $T = 5$ and in the initial configuration only the blue and orange lights are turned on. There are four more rotations, for $\phi \in \left\{ \frac{2\pi}{5}, \frac{4\pi}{5}, \frac{6\pi}{5}, \frac{8\pi}{5} \right\}$, all of them different.</p>
<pre>2 4 1 3</pre>	1	<p>In this example, $T = 4$ and in the initial configuration only the red and cyan lights are turned on. There are three more rotations, as shown below. However, only the first rotation (for $\phi = \frac{\pi}{2}$) is different. The second rotation (for $\phi = \pi$) is the same as the initial configuration, because the same lights are turned on. Also, the third rotation (for $\phi = \frac{3\pi}{2}$) is the same as the first rotation.</p>
<pre>6 12 0 1 3 6 7 9</pre>	5	<p>The initial configuration is shown in the image. There are five different rotations, for $\phi \in \left\{ \frac{\pi}{6}, \frac{\pi}{3}, \frac{\pi}{2}, \frac{2\pi}{3}, \frac{5\pi}{6} \right\}$.</p>

Magical Stones I

Time limit: 2480 ms

Memory limit: 264 MB

You are practicing your alchemy skill over a pile of magical stones. A magical stone has N possible states numbered from 1 to N . In the beginning, you have exactly one magical stone that is in each state i .

You know one magic spell. When you cast the spell, a stone that is in state i will transform into a stone in state S_i . Whenever two stones are in a same state i , they will purify each other, and combine into a single more powerful stone in state i . Multiple stones in a same state will combine at the same time.

You would like to obtain exactly K magical stones in the end. What is the minimum number of spells you have to cast to achieve that? Since you are not sure which K best satisfies your needs, you are going to answer this question for many choices of K .

Standard input

The input has a single integer N on the first line.

The second line has N integers. The i -th is S_i .

The next line has a single integer Q .

Each of the next Q lines has a single integer K as a query, for which you need to determine the minimum number spells required to obtain exactly K magical stones.

Standard output

For each query, output the minimum number of spells required on a single line. If it is impossible to obtain exactly K stones, output -1 .

Constraints and notes

- $2 < N \leq 10^5$
- $1 < S_i \leq N$. It is possible that $S_i = i$.
- $1 \leq Q \leq 10^5$
- In all queries $1 \leq K < N$
- For 50% of the test data, $N \leq 1\,000, Q \leq 1\,000$.

Input	Output	Explanation
<pre> 5 3 3 2 3 1 4 3 2 1 4 </pre>	<pre> 1 2 -1 -1 </pre>	<p>Initially, you have one stone in each of the 5 states $\{1, 2, 3, 4, 5\}$. When you cast the first spell, the stone state in 3 transforms to state 2 ($S_3 = 2$). The stone in state 5 transforms to state 1 ($S_5 = 1$). The stones in state 1, 2, 4 transform to state 3 ($S_1, S_2, S_4 = 3$). They purify each other and become one stone in state 3. Therefore after one spell, you will have three stones in states $\{1, 2, 3\}$. After a second spell, you will have two stones in states $\{2, 3\}$. You will not be able to combine the last two stones into one. You also cannot obtain four stones.</p>

Parkour

Time limit: 3080 ms

Memory limit: 264 MB

Catom is practicing parkour in the scaffolding of the new building in the city. The floor of the scaffolding is a grid. The cell in the x -th position to the right and the y -th position to the top has coordinates (x, y) . Each cell should contain a floor tile. But as the building is not finished, there are only N cells numbered from 1 to N that currently contains a floor tile at positions $(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$. These tiles are glued to the floor and called *stable tiles*.

In the entrance of the building there are k floor tiles waiting to be put on the floor. These tiles are called *movable tiles*. The movable tiles are very heavy and can only be moved using a crane. Catom has a remote control for the crane but it is very complicated to operate. Catom can thus only use the crane to move a movable tile when he is on a stable tile, because he needs the balance. When a movable floor tile is placed on a cell, it does not become glued to it, so it may be moved again. In other words, Catom can replace the movable floor tiles wherever he likes when standing on a stable tile.

Parkour is the art of getting from one point to another in a complex environment. Catom starts at stable tile s and wishes to reach stable tile t under the following constraints:

- Catom can only step on floor tiles on his way.
- Catom can move from one floor tile to another only if they share a common edge.
- Catom can only move the movable tiles, and only when he is on a stable tile.

Help Catom complete his parkour training by determining for some (s, t, k) values whether or not Catom can go from stable tile s to stable tile t . You need to answer Q such queries $(s_1, t_1, k_1), (s_2, t_2, k_2), \dots, (s_Q, t_Q, k_Q)$.

Standard input

To reduce the size of the input file, not all input data will be given explicitly. Instead, a set of parameters will be provided to generate the data.

The input has four integers N, Q, S_Q, M_k on the first line. S_Q is the number of queries that are explicitly given. M_k is a modulus used for generating the data.

The following N lines each have two integers as the position of one stable tile. The i -th line has x_i and y_i .

The next line has nine integer parameters $A_s, B_s, C_s, A_t, B_t, C_t, A_k, B_k, C_k$. These are followed by S_Q lines that each contain three integers as one query. The i -th line has s_i, t_i, k_i . The remaining queries with $S_Q < i \leq Q$ are generated using the following rules:

- $s_i = (A_s \cdot s_{i-1} + B_s \cdot s_{i-2} + C_s) \bmod N + 1$
- $t_i = (A_t \cdot t_{i-1} + B_t \cdot t_{i-2} + C_t) \bmod N + 1$
- $k_i = (A_k \cdot k_{i-1} + B_k \cdot k_{i-2} + C_k) \bmod M_k$

Standard output

Let the answer to the i -th query be R_i . If Catom can reach stable tile t from stable tile s then $R_i = 1$, and otherwise $R_i = 0$. Output a single integer $\sum_{1 \leq i \leq Q} R_i \cdot 2^i \bmod (10^9 + 7)$, which is the combined answers to all queries.

Constraints and notes

- $1 \leq N \leq 10^5$
- $1 \leq Q \leq 10^7$
- $2 \leq S_Q \leq Q$
- $1 \leq M_k \leq 10^9$
- $0 \leq x_i, y_i < 10^9$ for $1 \leq i \leq N$, all (x_i, y_i) are unique.
- $1 \leq s_i, t_i \leq N, 0 \leq k_i < M_k$ for $1 \leq i \leq Q$
- $0 \leq A_s, B_s, C_s, A_t, B_t, C_t, A_k, B_k, C_k \leq 10^9$

- For 25% of the test data, $N, Q \leq 500$.
- For another 25% of the test data, $N, Q \leq 10^5$ and $1 \leq M_k \leq 4$.
- For another 25% of the test data, $N, Q \leq 10^5$.

Input

```

3 2 2 100
0 0
2 5
5 3
0 0 0 0 0 0 0 0
1 3 6
2 1 5

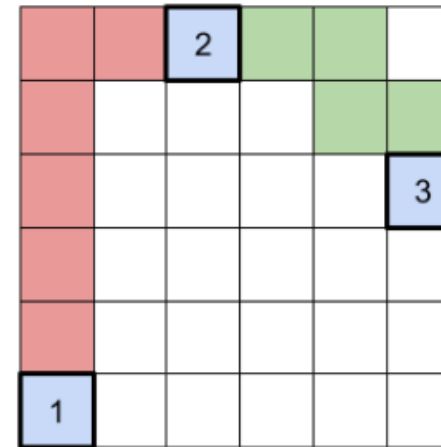
```

Output

```
2
```

Explanation

All stable tile positions and queries are explicitly given. The three stable tiles are illustrated below in blue.



For the query $s_1 = 1, t_1 = 3, k_1 = 6$ the answer should be $R_1 = 1$, because Catom can put the 6 movable floor tiles in a path from stable tile 1 to stable tile 2 (shown in red), go to stable tile 2, then move 4 of the movable tiles to connect stable tile 2 and stable tile 3 (shown in green) and finally get to stable tile 3.

For the query $s_2 = 2, t_2 = 1, k_2 = 5$ the answer should be $R_2 = 0$ since Catom can not reach stable tile 1 with fewer than 6 movable tiles.

The answer is therefore

$$(R_1 \cdot 2^1 + R_2 \cdot 2^2) \bmod (10^9 + 7) = (1 \cdot 2 + 0 \cdot 4) \bmod (10^9 + 7) = 2$$

The answer is therefore

$$(R_1 \cdot 2^1 + R_2 \cdot 2^2) \bmod (10^9 + 7) = (1 \cdot 2 + 0 \cdot 4) \bmod (10^9 + 7) = 2$$

```
3 7 2 9
0 0
2 5
5 3
7 8 10 3 2 1 5 4 59
1 3 6
2 1 5
```

242

Just 2 queries are explicitly given. The three stable tiles are the same as in test case #1.

The 7 queries are

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

$R_1, R_4, R_5, R_6, R_7 = 1$

$R_2, R_3 = 0$

The answer is therefore

$$\begin{aligned} & (R_1 \cdot 2^1 + R_2 \cdot 2^2 + \dots + R_7 \cdot 2^7) \bmod (10^9 + 7) \\ &= (1 \cdot 2 + 0 \cdot 4 + 0 \cdot 8 + 1 \cdot 16 + 1 \cdot 32 + 1 \cdot 64 + 1 \cdot 128) \bmod (10^9 + 7) \\ &= 242 \end{aligned}$$

Non-Overlapping Palindromes

Time limit: 3680 ms
Memory limit: 264 MB

Alice often likes to play with **palindromic** strings. Given a string S , she wants to find two non-empty palindromic **substrings** that are not overlapping. What is the maximum sum of lengths of these two palindromic substrings?

Standard input

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

Each of the next T lines gives one test case with a single string S .

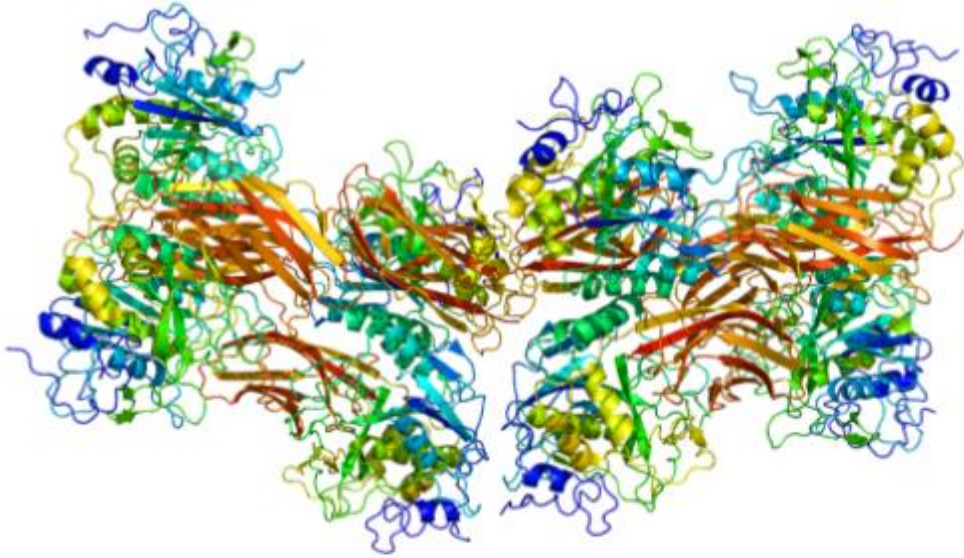
Standard output

For each test case, output a single line with the maximum sum of lengths.

Constraints and notes

- $1 \leq T \leq 10$
- S contains between 2 and 10^5 lowercase English letters.
- A string is palindromic if we can obtain the same string by reversing it. For example, `abcba`, `abba`, `a` are palindromic, and `abc` is not palindromic.

Input	Output	Explanation
3 xabcbayabbaz abcbaabc abcba	9 7 4	<code>xabcbayabbaz</code> contains substrings <code>abcba</code> and <code>abba</code> that are not overlapping. Their length sum is $5 + 4 = 9$. <code>abcbaabc</code> contains substrings <code>a</code> and <code>cbaabc</code> that are not overlapping. Their length sum is $1 + 6 = 7$



Hint: I gave no one real evidence. Furin is not dielectric; and not a great random acoustic magnet; oxidized filovirus; towards human enzyme; this is to lure enigma.

Standard input

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

The next T lines each contain a string of three printable ASCII characters representing one test case.

Standard output

For each test case, print the output of the given program on a single line.

Constraints and notes

- $1 \leq T \leq 10$
- A printable ASCII character has an unsigned 8-bit integer value between 33 and 126.

Input**Output**

```
4
xxx
XXX
***
<=>
```

```
12
90
44
4
```

```
2
0>?
dkx
```

```
264
884
```

Molecules

Time limit: 1280 ms

Memory limit: 264 MB

As you may remember, in the IEEEXtreme 10.0 there was a challenge called Counting Molecules. In that challenge, you had a machine that counted the number of molecules in a cup of soda which contains distilled water (H_2O), carbon dioxide (CO_2), and glucose ($C_6H_{12}O_6$). Given a cup of sample liquid, the machine reported the number of atoms of carbon, hydrogen, and oxygen as three integers C , H , and O respectively. You were asked to determine if the given number of atoms was consistent with a mixture containing only water, carbon dioxide, and glucose molecules. The answer could be `Error` when it was impossible to have a mixture of only water, carbon dioxide, and glucose molecules with the reported numbers of atoms.

In this challenge, you have the same machine that reports C , H , and O for a given liquid sample. We ask you to find the smallest number of atoms that needs to be added and/or discarded so that it is possible that the sample is a mixture of only water, carbon dioxide, and glucose.

Standard input

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

Each of the next T lines has three integers C , H , and O giving one test case.

Standard output

For each test case output the minimum number of atoms that needs to be added and/or discarded so that the sample becomes possibly a mixture of only water, carbon dioxide, and glucose. When the given numbers of atoms can already form a mixture of only water, carbon dioxide, and glucose, output zero.

Constraints and notes

- $1 \leq T \leq 40$
- $0 \leq C, H, O \leq 10^6$
- It is allowed to discard all atoms. Zero atoms are considered a valid (empty) mixture.

Constraints and notes

- $1 \leq T \leq 40$
- $0 \leq C, H, O \leq 10^6$
- It is allowed to discard all atoms. Zero atoms are considered a valid (empty) mixture.

Input	Output	Explanation
<pre>4 2 2 2 126 482 255 0 200 0 0 2 1</pre>	<pre>2 1 100 0</pre>	<ul style="list-style-type: none">• Test case 1: Adding one oxygen atom and removing a carbon atom makes it possible to be a mixture of 1 water molecule and 1 carbon dioxide molecule.• Test case 2: 7 molecules of CO_2, 121 molecules of water and 20 molecules of glucose give $7 + 6 * 20 = 127$ carbon atoms, $2 * 121 + 12 * 20 = 482$ hydrogen atoms, and $2 * 7 + 1 * 121 + 6 * 20 = 255$ atoms of oxygen, yielding only one extra atom (carbon).• Test case 3: We may add 100 oxygen atoms to have 100 water molecules.• Test case 4: Without adding or discarding any atoms, a single water molecule may have the given atoms.

Mosaic Decoration II

Time limit: 1280 ms
Memory limit: 264 MB

Zapray lives in a big mansion that has many bathrooms. He wants to decorate one bathroom wall using mosaic tiles. The bathroom wall has a width of W inches and a height of H inches. All mosaic tiles have a width of A inches and a height of B inches. The mosaic tiles are placed on the wall with their sides aligned horizontally and vertically. The right/bottom side of each tile is aligned with the left/top side of another tile, except for the tiles on the boundaries of the wall. The mosaic tiles have directional decorative patterns on them, and they cannot be rotated or flipped. Since A may not divide W , and B may not divide H , the tiles on the boundaries of the walls may have to be cut to fit. One mosaic tile can be cut into multiple pieces to be placed along different boundaries of the wall.

Mosaic tiles are sold in piles. Zapray can purchase one pile of 10 mosaic tiles for M dollars. He also has to pay the worker C dollars per inch for making cuts on the mosaic tiles. What is the minimum total amount of money that Zapray needs to cover the entire bathroom wall with mosaic tiles?

Standard input

The input contains six integers on a single line: W, H, A, B, M, C .

Standard output

Output a single integer, the minimum amount of money in dollar that Zapray needs.

Constraints and notes

- $1 \leq W, H, M, C \leq 10^6$
- $1 \leq A \leq W, 1 \leq B \leq H$

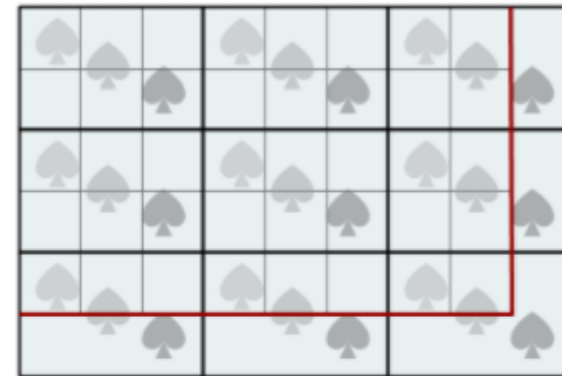
Input

8 5 3 2 100 3

Output

139

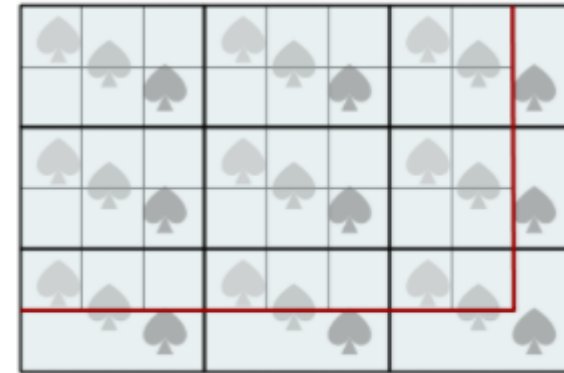
Explanation



Zapray needs 9 mosaic tiles and has to purchase one pile of tiles for 100 dollars. There are multiple ways to arrange the tiles. The figure shows one optimal way with a minimum total cost. Zapray needs to have the worker cut 5 of the tiles on the right and bottom boundaries. The total length of the cut is 13 inches and the cost of the cut is $13 \times 3 = 39$ dollars. The cut lines are illustrated in red.

5 2 3 2 100 3

106



Zapray needs 2 mosaic tiles and has to purchase one pile of tiles. There are multiple ways to arrange the tiles. The figure shows two ways. In the first way, Zapray places one full tile at the leftmost position. He makes one cut on the second tile and discards one-third of it. In the second way, Zapray places one full tile in the middle of the wall at 1.5 inches to the left boundary. He then makes cuts the second tile into three parts with a width of 0.5, 1, 1.5 inches. He discards the middle part, and places the right part (1.5 inches) to the leftmost of the wall. The cutting cost of the second way is higher than the first way. Thus the second way is not optimal.

98765 43210 1 1 777 1

331595290005

No tile cutting is needed.

The Last of Us

Time limit: 1280 ms
Memory limit: 264 MB

In this challenge, we will relive a nerve-racking journey of a popular game character named Ellie -- in an Xtreme way!

On the journey to her revenge, Ellie goes through a dangerous forest full of a type of monsters called *clickers*. A clicker was once a human, but contracted the dangerous Cordyceps fungus that turned people into monsters. A clicker is blind and has no sight. But it has an adaptive echolocation that can be used to identify its surroundings. A clicker is extremely sensitive to sound.

Ellie's journey will go through T areas. Each area is a 2D grid of cells. Each cell is either walkable (. or E) or blocked (#). The cell E is the exit of the area. There are clickers wandering in the area. Ellie is not carrying any ammo with her, and must bypass all the clickers and reach the exit of the area without engaging in a combat. Ellie is carrying B bricks with her, which can be thrown to distract the clickers.

- The *distance* between two cells $(x_1, y_1), (x_2, y_2)$ is $\max(|x_1 - x_2|, |y_1 - y_2|)$.
- The *absolute offset* between two cells $(x_1, y_1), (x_2, y_2)$ is $|x_1 - x_2| + |y_1 - y_2|$.
- Two cells $(x_1, y_1), (x_2, y_2)$ are *adjacent* if their distance is 1.
- The *surroundings* of a cell (x, y) consists of all the cells that have a distance no larger than 7 from (x, y) .
- The *vicinity* of a clicker consists of all the cells that have a distance no larger than 2 from the clicker's location. A clicker may hear any *sound* in its vicinity and become *alerted*.
- A cell is *empty* if it is walkable, does not contain a clicker, and not the exit of the map.
- The *shortest path* of a clicker between a pair of two walkable cells s, t are the shortest sequence of **walkable** cells (NOT empty cells) that connect s and t , excluding the exit E.

The game takes turns. Each turn starts with Ellie's move, followed by the move of each clicker in the area one by one.

On Ellie's turn, she may choose to:

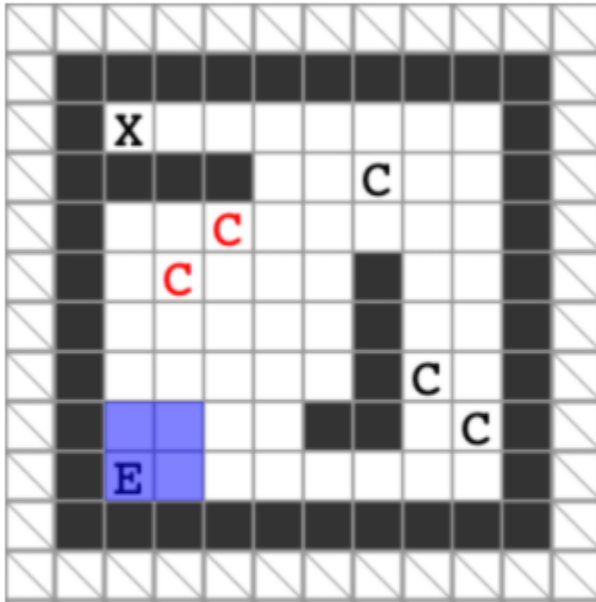
- Walk to an adjacent cell x . x must be an empty cell or the exit. Ellie succeeds if the cell is the exit of the area. Ellie makes a sound in x by walking into it.
- Throw a brick she carries to an empty cell x in her surroundings. The brick will crack and make a sound in x . A thrown brick cannot be used again.

On a clicker's turn, it follows these rules:

- If Ellie is standing in a cell that is adjacent to the clicker, the clicker will sense Ellie, grab and kill her immediately. Note that this takes the highest priority regardless of whether the clicker heard a sound or not.
- If a clicker is non-alerted, it will do nothing and stand still.
- If a clicker is alerted by a sound, it will walk to an adjacent cell on any shortest path to the cell x that produced the sound. If there are multiple cells on a shortest path, it will take the cell that has the smallest absolute offset to x . If there are still multiple such cells, the clicker will choose the cells in clockwise order starting from the cell on the top.
- If there is no path to cell x or the clicker is already in cell x , the clicker stands still. If a clicker heard multiple sounds, it will always trace the last sound.

Your goal is to write a program that helps Ellie navigate and exit the T areas of the game. We have provided a web game with an interface as shown below that allows you to play the T areas manually. This helps you better understand the game and develop a strategy. The T areas in the web game are exactly the same as the T areas on which your solution will be tested on the judging platform. An area can be identified by its area name given in the input.

[Download the web game](#)



Standard input

Each test file contains exactly one of the T areas. The input starts with one line of a single string, the area name. The second line of the input has a three integers R , C , and B . R and C indicate that the area size is R rows by C columns. B is the number of bricks Ellie is carrying.

The next R lines describe the area. Each line has a string of C characters. A walkable cell is `.`, a blocked cell is `#`, Ellie's location is `E`, the exit is `X`, and a clicker is `C`.

Standard output

Your solution should output a single integer K on the first line, the total number of actions Ellie needs to exit the area.

On each of the next K lines output one action:

- If Ellie chooses to walk to an adjacent empty cell, your program should print `w dx dy`, where (dx, dy) are the relative coordinates of an adjacent cell or Ellie's current cell. `dx` is $1(-1)$ if Ellie walks to the right (left), and `dy` is $1(-1)$ if Ellie walks to the top (bottom). It must satisfy that $\max(|dx|, |dy|) \leq 1$. Ellie may choose to stand still by `w 0 0`.
- If Ellie chooses to throw a brick, your program should print `b dx dy`, meaning that Ellie will throw a brick at the empty cell with relative coordinates (dx, dy) from her current location. It must satisfy $\max(|dx|, |dy|) \leq 7$.

Your solution must take no more than 10^5 actions, and satisfy $K \leq 10^5$.

Constraints and notes

- $10 \leq R, C \leq 500$
- $0 \leq B \leq 140$
- There is exactly one exit in each area. All walkable cells are reachable from the exit. It is guaranteed that there exists a way to reach the exit within 10^5 actions.
- The clickers in the web game behave exactly the same as they do on the judging platform. Therefore if a sequence of actions works for an area in the web game, it will also work for that area on the judging platform.
- Any invalid action will receive *Wrong Answer*. An invalid action can be walking to a blocked cell, throwing a brick at a non-empty cell, throwing a brick after Ellie's has used all the bricks, etc.
- The last action should be Ellie walking to the exit. Any action after Ellie reaches the exit will be considered invalid and receive *Wrong Answer*.
- Ellie succeeds immediately when she enters the exit, even if a clicker is adjacent to the exit.
- There can be at most one clicker in a walkable cell. An alerted clicker may stand still when all the adjacent cells on a shortest path are occupied by other clickers. A clicker cannot step onto the exit, which is considered not walkable for the clicker when it determines shortest paths.
- On the clickers' turn, clickers move one after another. The topmost and then leftmost clicker moves first.
- A clicker's movement does not produce a sound and does not alert other clickers.

Input	Output	Explanation
<pre> area-0-sample 10 10 2 ##### #X.....# ####..C..# #...C....# #.....#..# #C....#..# #.....#C.# #....##.C# #E.....# ##### </pre>	<pre> 14 b 1 5 w 1 0 w 1 0 w 1 1 w 1 1 b 3 5 w 0 1 w 0 1 w 0 1 w 0 1 w -1 1 w -1 0 w -1 0 w -1 0 </pre>	<p>You may interactively enter the example sequence of actions in the web game and view how Ellie can exit the area by these actions. Choose <code>sample</code> as the area in the web game. Ellie distracts the clickers by luring them to the boundaries of the area, opening a path in the middle leading to the exit.</p>

ARM Constant Multiplication

Time limit: 1280 ms

Memory limit: 264 MB

In this challenge, we will explore how a compiler for a 32-bit ARM processor may implement multiplication by an unsigned integer constant without using a multiply instruction, which would consume relatively a lot of energy. You are allowed to use the four instructions `MOV`, `ADD`, `SUB`, and `RSB` operating on registers where the last register may be logically shifted left or right (`LSL` or `LSR`). The allowed instructions have the following syntax:

- `MOV Rd, Ra, LSL/LSR #i` : Move the shifted value of `Ra` to `Rd`.
- `ADD Rd, Rb, Ra, LSL/LSR #i` : Add the shifted value of `Ra` to `Rb`, and save the result in `Rd`.
- `SUB Rd, Rb, Ra, LSL/LSR #i` : Subtract the shifted value of `Ra` from `Rb`, and save the result in `Rd`.
- `RSB Rd, Rb, Ra, LSL/LSR #i` : Subtract `Rb` from the shifted value of `Ra`, and save the result in `Rd`. `RSB` stands for Reverse SuBtract.

`Rd` is the destination register where the result of the instruction is saved. `Rb` and `Ra` are operand registers, and their values do not change during the instruction, though `Rd` may be the same as `Rb` and/or `Ra` in which case the value of the register may change. The value of `Ra` is shifted left (`LSL`) or right (`LSR`) by `i` bits before it is used by the instruction. That is, `Ra LSL #i` means the value of `Ra << i`, and `Ra LSR #i` means the value of `Ra >> i`. For `LSL`, the value of `i` must be between 0 and 31 inclusive. For `LSR`, the value of `i` must be between 1 and 32 inclusive. Note that the range of `i` is different for `LSL` and `LSR`. This is because the shifting amount `i` must fit into 5 bits in an instruction. All registers store 32-bit integers. Bits are discarded when shifted outside the 32-bit range. If you logically shift right by 32 (`LSR #32`), you end up with 0 in the register. Overflow bits are discarded in addition, and underflow bits are discarded in subtraction.

For each test case you will be given a single constant unsigned integer multiplier `C`. You should return a sequence of ARM instructions that implement the multiplication by this constant multiplier. Assume the multiplicand is in the register `R0`, and the product should end up in the same register `R0` afterwards. That is, if the value of the register `R0` is `X` in the beginning, its value should become `C · X` truncated to 32 bits in the end. You are allowed to use 12 additional registers `R1`, `R2`, ... `R12` to save any intermediate results. The values in `R1`, ... `R12` are initially all zeros. You are asked to minimize the number of instructions so that the multiplication can be executed as quickly as possible.

Standard input

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

Each test case has a single line with one unsigned 32-bit integer, the constant unsigned integer multiplier `C`.

Standard output

For each test case, output a sequence of instructions to perform the multiplication, one instruction per line. Print a single line with `END` to mark the end of the output for each test case. Each instruction should be one of the four classes given above and follow the provided formats. Note that the shift operation is required and you should set `i` to zero for `LSL` to use the value of `Ra` without any shift.

Scoring

For each test case, your solution will be scored based on k , the number of instructions used ($k \geq 0$). It can be shown that for any C , it is possible to implement the multiplication using no more than 32 instructions. Your submission will get a score of $(1.0 - k/33)^2$ if it correctly performs multiplication and satisfies $k \leq 32$. A submission that performs multiplication incorrectly or has $k > 32$ will receive zero points.

A test file will only receive a positive final score if **every** test case in the test file receives a positive score, in which case the final score of the test file is the average of the scores your solution receives for all the test cases in this test file times 5.

Constraints and notes

- $T = 100$ for all the hidden test files.
- Each C is an unsigned 32-bit integer.

Input	Output	Explanation
1 8	MOV R0, R0, LSL #3 END	R0 is shifted to the left by 3 bits, getting $R0 \ll 3$. This equals $8 * R0$ and is the product saved back to R0.
1 9	ADD R0, R0, R0, LSL #3 END	The ADD instruction adds 8 times R0 to R0, getting $9 * R0$, and saves it back to R0.
2 115 105	ADD R1, R0, R0, LSL #1 SUB R1, R1, R0, LSL #4 ADD R0, R1, R0, LSL #7 END RSB R1, R0, R0, LSL #2 MOV R0, R0, LSL #5 ADD R0, R0, R1, LSL #0 RSB R0, R0, R0, LSL #2 END	Let the initial value in R0 be X . The ADD instruction adds $2X$ from R0, LSL #1 and X from R0, getting $3X$ and saves it to R1. The SUB instruction gets $R1 - R0 * 16$ which is $3X - 16X = -13X$ and saves it in R1. The last ADD instruction obtains $R1 + 128 * R0$ which is $-13X + 128X = 115X$.

Note that in the last test case you can do better using only two instructions:

- `RSB R0, R0, R0, LSL #4`
- `RSB R0, R0, R0, LSL #3`

The first `RSB` places $15 * R0$ in `R0`. The second `RSB` calculates $7 * R0$. Before the second instruction, `R0` has $15X$, and therefore the second `RSB` will give us $105X$ in `R0` as a result.

Game of Life 2020

Time limit: 1280 ms

Memory limit: 264 MB

To help verify the May 2020's Ponder This** challenge, [May 2020 Ponder This](#), you need to write a simulator for a special version of Conway's game-of-life. In this game the board is cyclic (torus): the top of the board is connected to the bottom of the board and the left of the board is connected to the right of the board. We count the neighbors along common edges, so there are only 4 neighbors instead of the 8 neighbors found in the standard game-of-life.

Given the update rules for the empty cells and the live cells, and knowing the initial board, determine the final board state after a number of generations of the game.

Standard input

The input has the two update rules on the first line, separated by a semi-colon `;`. Each rule is given with exactly 5 bits of zeroes and ones. Empty cells should be updated using the first rule, and live cells should be updated using the second rule.

For empty cells, a `1` in the i -th bit of the rule sequence means that a live cell shall be born if exactly $i - 1$ of its 4 neighbors are alive.

Otherwise, the cell stays empty. For live cells, a `1` in i -th bit in the rule sequence means that a live cell shall stay live if exactly $i - 1$ of its 4 neighbors are also alive. Otherwise, the live cell becomes empty.

The second line of the input contains two integers N and M , indicating that the board is of size $N \times N$ and the number of generations to simulate is M .

The next N lines each have N bits of zeroes and ones, giving the initial state of the board. A live cell is denoted by a `1`, and an empty cell is denoted by `0`.

Standard output

Output the final board state after M generations. The output should contain N rows with each row containing N bits of `0`s and `1`s.

Constraints and notes

- $3 \leq N \leq 25$
- $1 \leq M \leq 1\,000$

Input	Output	Explanation
<pre>00100;11000 3 100 000 010 000</pre>	<pre>000 010 000</pre>	<p>The rules are that an empty cell becomes live if and only if it has 2 neighbors; and a live cell stays if and only if it has zero or one neighbors. So the board stays the same even after 100 generations.</p>
<pre>01100;01100 4 1 0000 1000 1000 0000</pre>	<pre>1000 1101 1101 1000</pre>	<p>Cells are born on all sides of the existing two cells. Since the board is cyclic, the left of the cells is at the rightmost of the board.</p>
<pre>10000;01100 10 4 0000000000 0000000000 0000000000 0000000000 0000000000 0000100000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000</pre>	<pre>0000000000 0000000000 0010101000 0001010000 0010101000 0001010000 0010101000 0000000000 0000000000 0000000000</pre>	<p>An empty board with this rule will oscillate between empty and full board. The single interference in the middle "explodes" outwards.</p>
<pre>10000;01100 10 100 0000000000 0000000000 0000000000 0000000000 0000100000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000</pre>	<pre>0010101000 0101010101 1000100010 0101010101 1010101011 0101010101 1000100010 0101010101 0010101000 0101110100</pre>	<p>This is the same as the previous sample, but it runs for 100 generations.</p>

Social Distancing in Class

Time limit: 1280 ms
Memory limit: 264 MB

The School of Xtreme is considering reopening in-person class for the best learning experience. To maximize social distancing, the school wants to seat the students in a classroom such that the sum of squared distances between all pairs of students is maximized. There are N students in the class numbered from 1 to N . There are exactly $N - 1$ pairs of students who are best friends. The best friend relationships form a tree structure, so that if everyone keeps sharing a story with his/her best friends, the story will eventually be shared across the whole class. To keep the students happy, we need to keep everyone no farther than 1 meter from each of their best friends.

The classroom can be thought of as a Euclidean plane of infinite size, and the students are points on this plane. What is the maximum possible sum of squared distances between every pair of students, if they are seated optimally?

Standard input

The first line of input has a single integer N , the of the number of students.

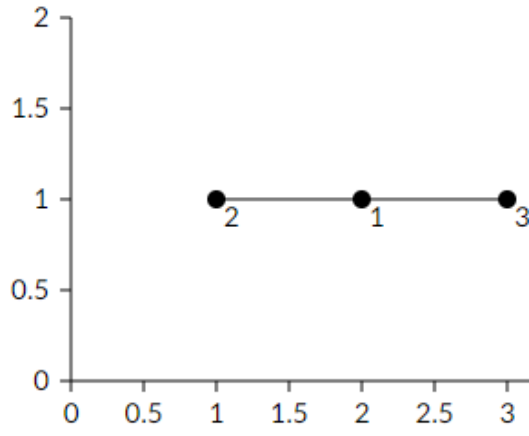
The next $N - 1$ lines each have a single integer, giving B_2, B_3, \dots, B_N , where B_i is the best friend of student i .

Standard output

Output a single line including the maximum possible sum of squares of distances. Your answer is considered correct if it has an absolute or relative error of 10^{-6} from the correct answer.

Constraints and notes

- $1 \leq N \leq 10^5$
- $B_i < i$
- It is fine for two students to be seated at the same location.
- For 30% of the test data, $N \leq 500$
- For 60% of the test data, $N \leq 5\,000$

Input	Output	Explanation
<pre>3 1 1</pre>	<pre>6</pre>	<p>There are 3 students. Both student 2's and student 3's best friends are student 1. We can let the three students sit in a row, with student 1 being in the middle and the other two students on the two sides at exactly 1 meter from student 1. The pairwise Euclidean distances in this optimal solution is $\{1, 1, 2\}$. Sum of their squares is 6. The figure illustrates this arrangement.</p> <p>Note that the absolute coordinates of the students do not matter. Only the relative distances between them affect the answer.</p> 
<pre>5 1 2 3 4</pre>	<pre>50.0</pre>	

Poker Game

Time limit: 1280 ms

Memory limit: 264 MB

Tommy is playing a special Poker game with a standard deck of 52 cards (Jokers excluded). There are four cards of each number: 2, 3, ..., 9, X, J, Q, K, A (X stands for 10). The suits of cards do not matter in this game, and cards of a same number are completely equivalent. Most winning hands in the original Poker game, such as flush and full house, are not available.

Tommy is dealt K cards as his *initial hand*. Then N community cards are dealt one by one. At any time, Tommy's *hand* is Tommy's initial hand cards combined with the community cards. Tommy scores points if his hand meets the following categories:

- Pair: Two cards of a same number give 1 point.
- Three of a kind: Three cards of a same number give 20 points.
- Four of a kind: Four cards of a same number give 1760 points.

For each unique card number, Tommy receives the highest points among these three. Tommy's final points are the sum of points he receives for each unique card number. For example, if Tommy's hand has a pair of 2s, a pair of 8s, three As, and four Ks, he scores $1 + 1 + 20 + 1760 = 1782$ points.

Tommy is not good at this Poker so he plays with a fixed simple strategy: Once a community card is dealt, if his points increase, he will raise the bet. Otherwise, he checks. No raise is allowed before the first community card is dealt.

You are watching Tommy playing this game, and you saw whether Tommy raised after each community card is revealed. You cannot help but wonder, what initial hand can Tommy possibly have?

Standard input

The input begins with two integers N and K on the first line.

The next line has a string of length N describing the community cards dealt in order.

The third line has a string of length N describing Tommy's actions after each community card. The i -th character is `y` if Tommy raised after the i -th community card, or `n` otherwise.

Standard output

Output a single line with the initial hand Tommy may have. Output the cards in the hand in sorted lexicographical order. If there are multiple possible hands, output the hand that gives the highest score. If there are still multiple hands, output the hand that is lexicographically the smallest.

The lexicographical order is defined over card numbers: $2 < 3 < \dots < 9 < X < J < Q < K < A$. For example, a hand with two 7s, one 3, one A and one X should be printed as `377XA`. The hand `377XA` is lexicographically smaller than `377KA`.

If Tommy's actions contradict with his assumed strategy, output a single word `impossible`.

Constraints and notes

- $2 \leq K \leq 9$
- $1 \leq N \leq 52 - K$
- The community cards present a valid deck of cards. For example, there will not be five cards of a same number.

Input	Output	Explanation
9 2 AA9Q3776J nyynnnynn	99	Tommy did not raise after the first A indicates that Tommy does not have A. Tommy raised after the second A because he got a pair and his score increased. Tommy raised immediately after the first 9 thus he must have a 9 in his initial hand. His other card cannot be uniquely determined. But that card cannot be A, Q, 3, 7, 6, J, so there are seven possibilities remaining: 29, 49, 59, 89, 99, 9X, 9K. Among them 99 gives the highest 22 points.
3 2 234 YYY	impossible	Tommy raised immediately after the first 2, 3, 4. This indicates that Tommy has 2, 3, 4 in his initial hand. However only two cards are dealt for the initial hand. Therefore this contradicts Tommy's strategy.
3 2 XXX nyn	impossible	Tommy's score must increase after the third X is dealt. Tommy must raise according to his strategy but he didn't.
8 3 234567AK nnnnnnyy	KKA	Two hands give a highest score of 21 points: KKA and KAA. KKA is lexicographically smaller.

Magical Stones II

Time limit: 1280 ms

Memory limit: 264 MB

You are practicing your alchemy skill over a pile of magical stones. A magical stone has N possible states numbered from 1 to N . In the beginning, you have exactly one magical stone that is in each state i .

You know two magic spells: the *white* spell, and the *black* spell. When you cast the white spell, a stone that is in state i will transform into a stone in state W_i . When you cast the black spell, a stone that is in state i will transform to a stone in state B_i . Whenever two stones are in a same state i , they will purify each other, and combine into a single more powerful stone in state i . Multiple stones in a same state will combine at the same time.

You would like to obtain one single magical stone that is the purest and the most powerful of all. You will cast a sequence and black and white spells and combine all the N stones you have into one. The final stone can be in any state. Is this ever possible?

Standard input

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

Each test case has single integer N on the first line. The next two lines each have N integers. The first line gives W_1, \dots, W_N . The second line gives B_1, \dots, B_N .

Standard output

Output one line for each test case.

If it is possible to combine all the N stones into one, output a string that is the sequence of magic spells you will cast. Output `w` for a white spell and `b` for a black spell. If there are multiple possible sequences, you may output any of them. You do not need to find a shortest sequence of spells. However since you don't want to spend the whole 24 hours casting spells, you must output a sequence of no more than 10^6 spells.

If it is impossible to combine all the stones into one, output a single word `impossible`.

Constraints and notes

- $1 \leq T \leq 30$
- $2 \leq N \leq 100$
- $1 \leq W_i, B_i \leq N$. It is possible that $W_i = i$ or $B_i = i$.
- It can be shown that if it is possible to combine all stones into one, then there exists a sequence of no more than 10^6 magic spells to do so.
- Your answer is considered correct as long as you have a single stone at the end of the spell sequence. That is, it is allowed to cast redundant spells after successfully combining the stones into one.

- For 33% of the test data, $N \leq 20$

Input	Output	Explanation
<pre> 3 4 2 3 4 1 3 3 2 3 5 1 2 3 4 5 2 3 4 5 5 4 2 3 4 1 4 1 2 1 </pre>	<pre> BWWB BBBBWW impossible </pre>	<p>Case 1: The states the stones are in after each spell are:</p> <ul style="list-style-type: none"> • At the beginning: $\{1, 2, 3, 4\}$ • After a black spell: $\{2, 3\}$ • After a white spell: $\{3, 4\}$ • After a white spell: $\{1, 4\}$ • After a black spell: $\{3\}$ <p>Case 2: After four black spells you will obtain a single stone in state 5. It is okay that you cast some redundant spells after that, as long as the total number of spells does not go above 10^6.</p> <p>Case 3: The white spell does not allow you to combine any stones. Using only black spells will be able to combine the stones that are initially in states 1, 3 and initially in states 2, 4, but not all the states.</p>

Rescue Mission

Time limit: 3680 ms
Memory limit: 264 MB

A group of Xtreme soldiers are fighting a tough war but are unfortunately trapped within the enemy territory. But don't worry, they managed to find N hideouts along a long battle line. The hideouts are numbered 1 to N . Initially there are S_i soldiers at hideout i .

There is a safe rendezvous location. Each hideout has one path to the rendezvous location. However, since the enemies are heavily patrolling the area, that path between the rendezvous location and hideout i cannot be taken unless the weather around hideout i is foggy.

You are planning a rescue mission to safely evacuate these soldiers in the next D days. You are able to forecast that on day i , the hideouts numbered $L_i, L_i + 1, \dots, R_i$ will have foggy weather, and will be able to gather at the rendezvous location. You will send a vehicle that can evacuate V_i soldiers. The remaining soldiers must go back to the hideouts. The soldiers do not necessarily need to go back to the hideout where they came from. Instead, they can go to any hideout with a number between L_i and R_i . Each hideout may have an arbitrary number of soldiers at any time, including zero.

If you coordinate the movements of the soldiers carefully, what is the maximum total number of soldiers you can evacuate?

Standard input

There is a single integer N on the first line, the number of hideouts. The second line has N integers. The i -th integer is S_i .

The next line has a single integer D , the number of days. Each of the next D lines has three integers L_i, R_i , and V_i . They indicate that on day i hideouts $L_i, L_i + 1, \dots, R_i$ will have foggy weather, and you will send a vehicle to the rendezvous location that can evacuate V_i soldiers from these hideouts.

Standard output

Output the maximum total number of soldiers you can evacuate.

Constraints and notes

- $1 \leq N \leq 10^5$
- $0 \leq S_i \leq 10^4$
- $1 \leq D \leq 5\,000$
- $1 \leq L_i \leq R_i \leq N$
- $1 \leq V_i \leq 10^9$

- For 30% of the test data, $D \leq 50$ and $N \leq 50$.
- For 60% of the test data, $D \leq 50$.

Input	Output	Explanation
<pre> 4 5 4 3 2 4 1 2 4 1 1 3 2 4 1 3 3 4 </pre>	<pre> 12 </pre>	<p>At the beginning, the number of soldiers at the hideouts are $[5, 4, 3, 2]$. On day 1, there are 9 soldiers from hideout 1 and 2 that can be evacuated. The vehicle takes 4, and the 3 of the 5 remaining soldiers can go to hideout 1 to wait for the vehicle on day 2. The other 2 remaining soldiers go to hideout 2. The number of soldiers at the hideouts are therefore $[3, 2, 3, 2]$. After day 2, the numbers become $[0, 2, 3, 2]$. On day 3, we can evacuate one soldier from hideout 2, and let the other soldier there go to hideout 3. The numbers of soldiers at the hideouts become $[0, 0, 4, 2]$. On the last day, the 4 soldiers at hideout 3 will be evacuated.</p>
<pre> 3 7 8 7 6 1 2 1 2 3 1 3 3 9 2 3 1 1 2 1 1 1 9 </pre>	<pre> 22 </pre>	<p>At the beginning, we have $[7, 8, 7]$ soldiers. In the first two days, two soldiers can be evacuated, and at the same time all soldiers can move to hideout 3, getting $[0, 0, 20]$ by the end of day 2. On day 3 we can evacuate 9 soldiers, getting $[0, 0, 11]$. Then on day 4 and 5 two soldiers can be evacuated, and at the same time all soldiers can move to hideout 1, getting $[9, 0, 0]$. On the last day all the remaining soldiers can be evacuated.</p>

Restaurant Reopening

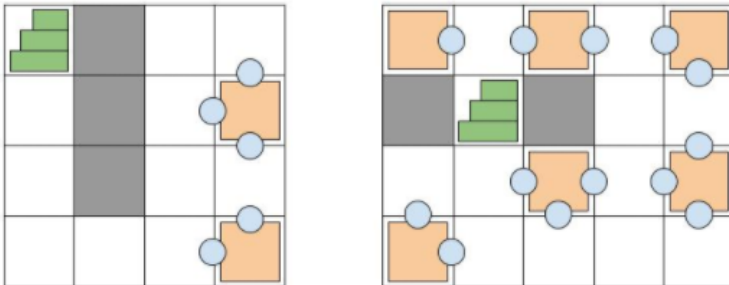
Time limit: 6080 ms
Memory limit: 264 MB

Many restaurants have been closed due to the global pandemic, including Cafe Xtreme. Now that the reopening is ongoing while indoor dining is still restricted, the manager of Cafe Xtreme thinks that it would be a good opportunity to redesign the restaurant floor layout to better use its space. Cafe Xtreme is big and it has multiple floors. Each floor of the restaurant can be viewed as an $R \times C$ grid with R rows and C columns. Two cells are adjacent if they share an edge. A cell may be empty, contain a wall, or contain the stairs. There is exactly one cell with the stairs per floor, which is the entrance and exit.

The manager will place dining tables in some empty cells of the floor. Each dining table occupies an entire empty cell. Dining chairs are placed in the adjacent empty cells of a dining table. A dining chair must be placed towards an edge of a dining table, and allows a customer to sit at the left, right, top, or bottom of the dining table. A dining table can therefore serve between one to four customers, depending on how many dining chairs are placed around it. A dining chair takes a negligible amount of space, so that multiple dining chairs that belong to different dining tables may be placed in a same empty cell. However, dining chairs cannot be placed in a cell that has a wall, a dining table, or the stairs.

A restaurant customer can walk through a cell as long as the cell does not contain a wall or a dining table. In particular, a customer may walk through a cell that contains his/her own dining chair or other customers' dining chairs (slipping through people's backs). All customers must be able to reach the stairs from their dining chairs so that they can enter and exit the restaurant.

The manager wants to know the maximum number of customers that each restaurant floor can accommodate.



Standard input

The first line has a single integer T , the number of floors to consider. This is followed by the description of T floors.

Each floor starts with two integers R and C on a single line.

The next R lines each have a string of length C . Each character in the string can be a dot `.` as an empty cell, a hash `#` as a wall, or a letter `s` as the stairs.

Standard output

For each floor, output the maximum number of customers the floor can accommodate on single line.

Constraints and notes

- $1 \leq T \leq 10$
 - $4 \leq R \cdot C \leq 100$
 - Each floor map has exactly one letter 'S', and at least two empty cells.
 - Before placing any dining tables and chairs, it is possible to walk to the stairs from every empty cell on a floor.
-
- For 37.5% of the test files, $R \cdot C \leq 20$.

Input	Output	Explanation
<pre>3 4 4 S#.. .#.. .#.. 4 5 #S#.. 2 3 #.# .S.</pre>	<pre>5 13 0</pre>	<p>The image illustrates an optimal layout for the first two floors in the sample test case. Each dining chair is shown as a blue dot, and each dining table is shown as an orange square. The walls are marked grey. It can be seen that every blue dot can move to the stairs via a sequence of white cells, which are either completely empty or contain only dining chairs.</p> <p>In the first floor, no customer can be seated in the first or the second column from the left, otherwise the path to the stairs will be blocked.</p>

Coin Collector

Time limit: 1280 ms
Memory limit: 264 MB

Zapray has been playing the game World of Warcraft Classic. In the game, he needs to collect coins to exchange for rewards from a tribe. There are 9 kinds of coins he could collect. He can turn in the following combinations of coins for rewards, as many times as he likes.

- Set I: `zulian` Coin $\times 1$, `razzashi` Coin $\times 1$, and `hakkari` Coin $\times 1$
- Set II: `sandfury` Coin $\times 1$, `skullsplitter` Coin $\times 1$, and `bloodscalp` Coin $\times 1$
- Set III: `gurubashi` Coin $\times 1$, `vilebranch` Coin $\times 1$, and `witherbark` Coin $\times 1$

Each completed set gives the same rewards. Zapray has figured out he wants N sets of coins for the rewards, thus he is going to buy the coins from other players via the Auction House. Each of the listings on the Auction House always consists a number of the same kind of coins with a buyout price.

Now Zapray needs your help to find out the minimum amount of money to complete N sets of coins.

Standard input

The first line of input consists two integers N and M .

Each of the next M lines describes a listing of coins on the Auction House with the name of the coin, quantity K as an integer and price P as an integer, separated by spaces. For example, `zulian 3 100` means 3 Zulian Coins are listed at a price of 100.

Zapray cannot partially buy a listing.

Standard output

If it is impossible, output a single line `impossible`. Otherwise, output the minimum amount of money Zapray has to spent on completing N sets of coins.

Constraints and notes

- $0 \leq N \leq 1000$
- $0 \leq M \leq 5000$
- $0 < K \leq 250$
- $0 < P \leq 10^6$
- Coin names in the input are case-sensitive, and each coin name is exactly one of the 9 coin names as in the statement.
- World of Warcraft Classic is developed and released by Blizzard Entertainment. Blizzard Entertainment does not endorse and has no involvement with IEEEXtreme 14.0 contest.

Input	Output	Explanation
<pre>10 8 Zulian 3 100 Sandfury 5 500 Skullsplitter 6 900 Bloodscalp 5 400 Razzashi 5 125 Hakkari 7 375 Witherbark 7 60 Zulian 4 800</pre>	<pre>3200</pre>	Zapray needs to buy all listings except for Witherbark Coins to complete 10 sets.

```
10 3
Zulian 100 100
Sandfury 100 100
Gurubashi 100 100
```

```
impossible
```

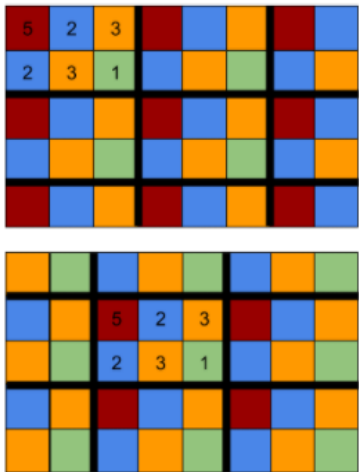
There is no possible ways to complete enough sets.

Mosaic Decoration III

Time limit: 1280 ms
Memory limit: 264 MB

Zapray lives in a big mansion that has many bathrooms. He wants to decorate one bathroom wall using mosaic tiles. The bathroom wall can be viewed as a grid of N rows and M columns of cells. Each mosaic tile covers exactly one cell on the wall. Zapray has a *base pattern* of mosaic tiles that has R rows and C columns. Each tile in the base pattern has some color labeled by a positive integer. Zapray will first choose one location on the wall to place the base pattern, and then he will repeat the base pattern horizontally and vertically to cover the entire wall.

In the example below, the wall has $N = 5$, $M = 8$, and the base pattern has $R = 2$, $C = 3$. The base pattern has tiles of four different colors labeled 1, 2, 3, 5. If Zapray chooses to place the base pattern at the top-left of the wall, the pattern will fill the wall as shown in the first image. If Zapray moves the base pattern to the right by two columns, and to the bottom by one row, the pattern will fill the wall as shown in the second image.



For any color labeled c , Zapray can purchase one mosaic tile of that color for c dollars. If Zapray chooses to place the base pattern at the best location and then fill the wall, what is the minimum total cost he has to pay to have the entire wall decorated by mosaic tiles?

Standard input

The input has four integers N, M, R, C on the first line.

The next R lines each have C positive integers, giving the mosaic tile colors in the base pattern.

Standard output

Output a single integer, the minimum total cost in dollars that Zapray needs to pay.

Constraints and notes

- $1 \leq N, M \leq 10^6$
- $1 \leq R \leq \min(50, N), 1 \leq C \leq \min(50, M)$
- All color labels are between 1 and 10^6 .

Input	Output	Explanation
<pre>5 8 2 3 5 2 3 2 3 1</pre>	98	<p>This test case is illustrated in the figure above. The second base pattern location gives the optimal solution that needs 4 tiles of color 5 (shown in red), 12 tiles of color 2 (shown in blue), 15 tiles of color 3 (shown in orange), and 9 tiles of color 1 (shown in green). The total cost is</p> $4 \times 5 + 12 \times 2 + 15 \times 3 + 9 \times 1 = 98$ <p>dollars.</p>

Coupon Codes

Time limit: 2480 ms
Memory limit: 264 MB

Brett is running a new online goodie shop. The business is booming and it has attracted tens of thousands of customers! Yet Brett is ambitious and he would like to further expand his customer base. He will run a campaign that provides discount coupon codes to new customers.

Brett will generate N coupon codes of the following format:

```
xxxx-xxxx-xxxx
```

where x is an uppercase letter `A-Z` or a digit `0-9`.

At first Brett thought he'll just randomly generate these codes. Soon he realizes that there is a serious issue - people tend to mistype their codes. Brett definitely does not want to accept a mistyped code, otherwise one customer's discount may be used by another customer!

Brett would like to know among the N coupon codes he generated how many pairs of them are *similar*. Two coupon codes are similar if their Hamming distance is exactly one. In other words, two coupon codes C_1, C_2 are similar if you can change C_1 into C_2 by modifying exactly one character in C_1 .

Standard input

The first line has a single integer N .

Then N lines follow. Each line has one coupon code.

Standard output

Output a single integer, the number of pairs of coupon codes that are similar.

Constraints and notes

- $2 \leq N \leq 10^5$
- All coupon codes are distinct, and are valid according to the format above.
- For 50% of the test data, $N \leq 100$

Input	Output	Explanation
<pre>6 WELC-OMET-OTHE IEEE-XTRE-ME14 AAAA-0000-A0A0 AAAA-0000-A0A1 AAAA-0000-A0AB AAAA-0000-ABAB</pre>	<pre>4</pre>	There are 6 codes. If they are labeled from 1 to 6, the similar pairs are (3, 4), (3, 5), (4, 5), (5, 6).