1901 SHANDONG UNIVERSITY

CIIC
国家信息通信国际创新园
CHINA INTERNATIONAL ICT INNOVATION CLUSTER

The 10th China Collegiate Programming Contest (Jinan)

October 27, 2024

	Problem	Time / Memory
A	The Fool	1s / 1024MB
B	The Magician	1s / 1024MB
C	The Empress	2s / 1024MB
D	The Emperor	2s / 1024MB
E	The Chariot	2.077s / 1024MB
F	The Hermit	2s / 1024MB
G	The Wheel of Fortune	7s / 1024MB
H	Strength	3s / 1024MB
I	The Hanged Man	3s / 1024MB
J	Temperance	2s / 1024MB
K	The Devil	7s / 1024MB
L	The Tower	3s / 1024MB
M	Judgement	1s / 1024MB

Total: 14 sheets, 28 pages

DO NOT OPEN THE PROBLEM SET UNTIL CONTEST STARTS.

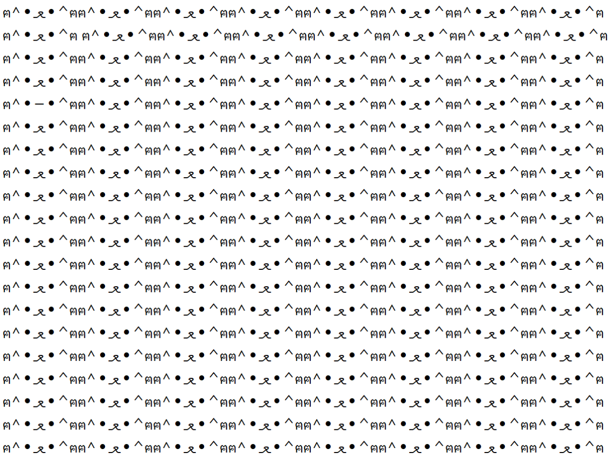
This page is intentionally left blank.

Problem A. The Fool

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

> *"This place tastes like many emotions. Excitement, anger, joy and... salty? Is salty a feeling?"*
>
> — Neeko, *The Curious Chameleon*

Hailing from a long lost tribe of vastaya, Neeko can blend into any crowd by borrowing the appearances of others, even absorbing something of their emotional state to tell friend from foe in an instant. No one is ever sure where — or *who* — Neeko might be, but those who intend to do her harm will soon witness her true colors revealed, and feel the full power of her primordial spirit magic unleashed upon them.



Source: https://bilibili.com/BV1ub421e7cd/

Neeko has infiltrated a grid of characters with dimensions $n \times m$, each cell in the grid is represented by a string of length k. She can mimic the other cells; however, Neeko's cell has *at least one character different* from the other cells while *all the other cells are the same*. Your task is to find Neeko.

Input

The first line of the input contains three integers n, m, k ($2 \le n, m \le 200, 1 \le k \le 10$).

The next n lines contain $m \cdot k$ characters each, consisting of visible ASCII characters from `!`(33) to `~`(126). The $((j-1) \cdot k + 1)$-th to the $(j \cdot k)$-th character in the i-th line represents the cell (i, j).

It's guaranteed that there is no extra space or line break, and the answer can be uniquely determined.

Output

Print two integers r, c, denoting the position of Neeko is in the r-th row and the c-th column.

Example

`standard input`	`standard output`

`3 5 3`
`QWQQWQQWQQWQQWQ`
`QWQQWQQWQQWQQWQ`
`QWQQWQQWQQWQQwQ`
	`3 5`

This page is intentionally left blank.

Problem B. The Magician

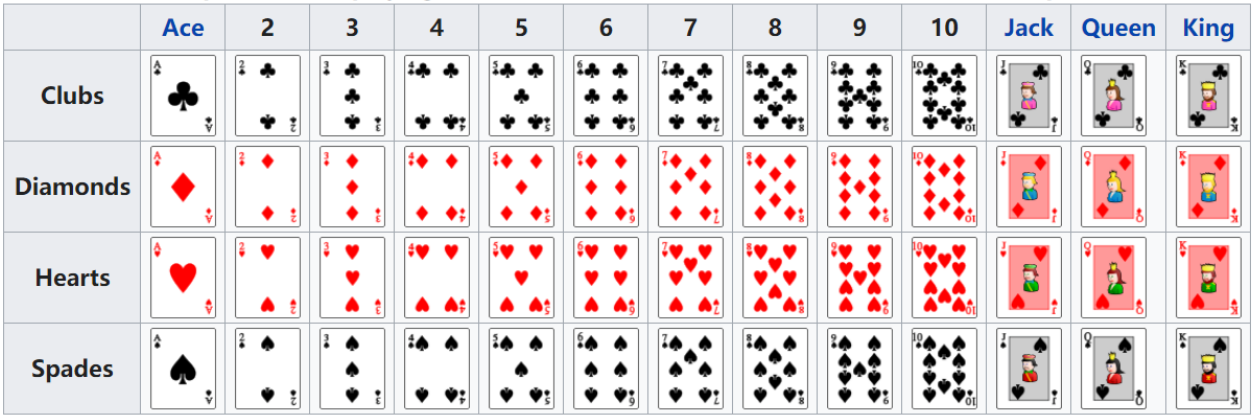
Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

> *"And now, the curtain rises."*
>
> — *????*

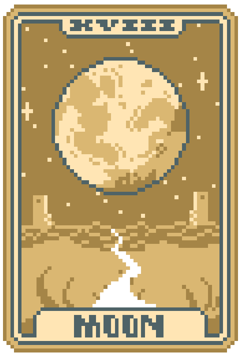
You are given some special Tarot cards from the Major Arcana, including *The Lovers*, *Death*, *The Star*, *The Moon*, *The Sun*, and *The World*, along with some other playing cards in hand. You have at most one of each kind of Tarot card, and each Tarot card has a unique ability that can alter the suits of other playing cards in your hand. Your task is to determine the maximum number of **flushes** that can be played by the playing cards in hand after using the given Tarot cards each at most once.

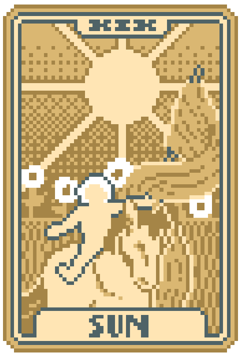
The playing cards are standard playing cards, shown as below.

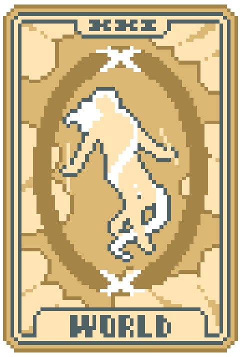
Example set of 52 playing cards; 13 of each suit: clubs, diamonds, hearts, and spades



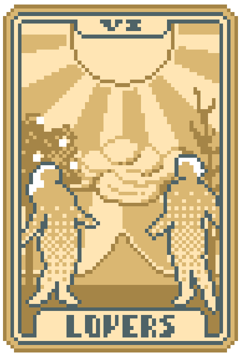
Here is a list of Tarot cards abilities.

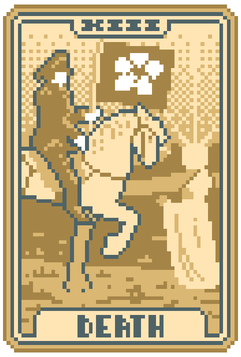






- *The Star*: Converts up to 3 selected playing cards to Diamonds (♢).
- *The Moon*: Converts up to 3 selected playing cards to Clubs (♣).
- *The Sun*: Converts up to 3 selected playing cards to Hearts (♡).
- *The World*: Converts up to 3 selected playing cards to Spades (♠).





- *The Lovers*: Converts 1 selected playing card into a *Wild Card* (⊛, can be used as any suit).
- *Death*: Select exactly 2 playing cards, replace one card with the copy of the other card (copy the suit, the rank, and the *Wild Card* status).

You can use the Tarot cards in any order. Each given Tarot card can be used at most once and can never be used at all. There is a special rule about *The Lovers*:

- Once a playing card has been converted into a *Wild Card* (using *The Lovers* or *Death*), it remains a *Wild Card* even after applying *The Star*, *The Moon*, *The Sun*, and *The World*;
- However, if *Death* is used to replace a *Wild Card* with a copy of another card that is not a *Wild Card*, the resulting card will not be a *Wild Card*.

A **flush** is a set of 5 playing cards, which could be considered a same suit: there is a suit (among Diamond, Club, Heart, and Spade) such that each of the 5 playing cards is either of this suit or is a *Wild Card*.

Playing a **flush** means that the 5 cards forming the **flush** are discarded from hand, all of which cannot be used in another **flush**. To the contrary of many card games including *Balatro*, you draw no new cards from the deck after playing cards.

Input

The input consists of multiple test cases. The first line contains a single integer T ($1 \le T \le 13$) — the number of test cases. The description of the test cases follows.

The first line contains an integer n ($1 \le n \le 52$), the number of playing cards in hand.

The second line contains n space-separated strings, each representing a playing card in hand. Each playing card is represented by two characters: one for rank and one for suit, where the suit is one of `D` (Diamonds), `C` (Clubs), `H` (Hearts), or `S` (Spades), and the rank is one of `2`-`9`, `T` (10, Ten), `J` (Jack), `Q` (Queen), `K` (King), or `A` (Ace).

The third line contains six space-separated integers $t_1, t_2, t_3, t_4, t_5, t_6$ ($0 \le t_i \le 1$), where:

- t_1 represents the number of *The Star*.
- t_2 represents the number of *The Moon*.
- t_3 represents the number of *The Sun*.
- t_4 represents the number of *The World*.
- t_5 represents the number of *The Lovers*.
- t_6 represents the number of *Death*.

It is guaranteed that the sum of n among T test cases does not exceed $104 = 52 \times 2$, and the playing cards are pairwise distinct in each test case.

Output

For each test case, print the maximum number of **flushes** that can be played in a single line.

Example

standard input	standard output
4 5 2H 3H 4H 5H 6D 1 1 1 1 0 0 5 2S 3S 4D 5C 6D 0 0 1 0 1 1 5 2S 3S 4D 5C 6D 0 0 1 0 1 0 13 AS 2S 3S 4S 5H 6H 7H 8H 9H TH JH QH KH 0 0 0 0 0 1	1 1 0 2

Note

In the first case, we can convert 6♢ into 6♡ using *The Sun*, and play 2♡3♡4♡5♡6♡ as a **flush**. This is not the only possible way; another possible way to play the same set of cards is:

- Convert 4♡5♡6♢ into 4♠5♠6♠ using *The World*;
- Convert 4♠5♠6♠ into 4♡5♡6♡ using *The Sun*.

In the second test case, one possible way to play a **flush** is:

- Convert 2♠3♠4♢ into 2♡3♡4♡ using *The Sun*;
- Convert 5♣ into 5⊛ using *The Lovers*;
- Replace 6♢ with a copy of 5⊛ using *Death*;
- Play 2♡3♡4♡5⊛5⊛.

This page is intentionally left blank.

Problem C. The Empress

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

Note: This is the inverse version of problem "The Emperor" with differences in constraints.

Capoo invented an interesting language named Push-Pop. This language is an interpreted language. The interpreter starts with an empty stack with infinite capacity and reads the first instruction of the custom program. There are only two kinds of instructions in this language:

- `POP a GOTO x; PUSH b GOTO y`

  If the top element of the stack is a, then pop the stack once and transfer the control flow to the x-th instruction (which means the next instruction will be the x-th).
  Otherwise, push an element b into the stack and transfer the control flow to the y-th instruction.

- `HALT; PUSH b GOTO y`

  If the stack is empty, halt the whole program after executing this instruction.
  Otherwise, push an element b into the stack and transfer the control flow to the y-th instruction.

Capoo wants to construct a Push-Pop program that halts after executing **exactly k instructions**. Due to the naive implementation of the interpreter, a program can contain **at most 64 instructions**.

Input

The only line contains a single integer k ($1 \le k \le 2^{31} - 1$, k is odd).

Output

The first line contains an integer n ($1 \le n \le 64$) denoting the number of instructions, and then follows n lines denoting the Push-Pop program. For each instruction, $1 \le a, b \le 128,\ 1 \le x, y \le n$ should hold.

It is guaranteed that a solution exists for given input.

Examples

standard input	standard output

`1`	`1`
	`HALT; PUSH 1 GOTO 1`

`5`	`5`
	`POP 1 GOTO 2; PUSH 1 GOTO 2`
	`HALT; PUSH 1 GOTO 3`
	`POP 1 GOTO 4; PUSH 2 GOTO 4`
	`POP 1 GOTO 2; PUSH 2 GOTO 4`
	`HALT; PUSH 99 GOTO 4`

Note

For the second example, instructions are: 1(`PUSH`), 2(`PUSH`), 3(`POP`), 4(`POP`), 2(`HALT`).

Key differences in constraints comparing to "The Emperor":

- $n \le 64$;
- The solution will always exists for given input;
- Program should halt in **exactly** k instructions, **not** to be considered modulo 998 244 353.

This page is intentionally left blank.

Problem D. The Emperor

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

Note: This is the inverse version of problem "The Empress" with differences in constraints.

Capoo invented an interesting language named Push-Pop. This language is an interpreted language. The interpreter starts with an empty stack with infinite capacity and reads the first instruction of the custom program. There are only two kinds of instructions in this language:

- `POP a GOTO x; PUSH b GOTO y`

  If the top element of the stack is a, then pop the stack once and transfer the control flow to the x-th instruction (which means the next instruction will be the x-th).
  Otherwise, push an element b into the stack and transfer the control flow to the y-th instruction.

- `HALT; PUSH b GOTO y`

  If the stack is empty, halt the whole program after executing this instruction.
  Otherwise, push an element b into the stack and transfer the control flow to the y-th instruction.

Capoo wants to upgrade the naive interpreter to deal with more instructions. Given a program of **at most** 1024 **instructions**, calculate the number of steps the program would execute before halting.

Input

The first line contains an integer n ($1 \le n \le 1024$), followed by n lines containing one instruction each. It is guaranteed that $1 \le a, b \le 1024$, $1 \le x, y \le n$ for each instruction.

Output

Print -1 if the program will never halt, or the number of instructions would execute, **modulo** 998 244 353.

Examples

`standard input`	`standard output`

`1`
`HALT; PUSH 1 GOTO 1`
	`1`

`5`
`POP 1 GOTO 2; PUSH 1 GOTO 2`
`HALT; PUSH 1 GOTO 3`
`POP 1 GOTO 4; PUSH 2 GOTO 4`
`POP 1 GOTO 2; PUSH 2 GOTO 4`
`HALT; PUSH 99 GOTO 4`
	`5`

`1`
`POP 1 GOTO 1; PUSH 1 GOTO 1`
	`-1`

Note

Key differences in constraints comparing to "The Empress":

- $n \le 1024$;
- The answer may not exist (program never halts), in which case report -1;
- For every program that will halt, print the result **modulo** 998 244 353.

This page is intentionally left blank.

Problem E. The Chariot

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2.077 seconds
Memory limit:	1024 megabytes

In the ever-changing and chaotic environment of Night City, uncertainty is the only certainty. For safe travel, edgerunners rely on the best technology. Despite their premium price and corporation background, Delamain AI armed cabs provide the most reliable mode of transportation.

Today, you need to make a lengthy journey across Night City. The Delamain cab's fare structure is as follows:

- The base fare is A eurodollars, covering the first X meters.
- For the next Y meters, the fare is B eurodollars per meter.
- Beyond $X + Y$ meters, the fare is C eurodollars per meter.

At any point, you may perform the following operation: stop and re-hail a cab. By doing so, you will immediately settle the cost of your previous cab, and the fare for the next one will be recalculated starting from the base fare. This operation can be performed for any number of times.

Now, you want to determine the minimum cost in eurodollars for traveling D meters using Delamain cabs.

Input

The input consists of multiple test cases. The first line contains a single integer T ($1 \le T \le 2077$) — the number of test cases. The description of the test cases follows.

The input contains six integers in a line: A, B, C, X, Y, D ($0 < A, B, C, X, Y, D < 10^{2077}$) in decimal representation, without leading zeros.

It is guaranteed that the sum of the number of digits of A among T test cases does not exceed `0x2077`. This constraint also applies individually to B, C, X, Y, D.

The value `0x2077` used in the constraint is a hexadecimal number, equal to the decimal number 8311.

Output

For each test case, output a single line containing the minimized cost in eurodollars for traveling D meters using Delamain's cabs.

It can be proven that the answer is a positive integer. Please print the integer in decimal form without leading zeros.

Example

standard input	standard output

`5`
`160 27 41 3 12 3`
`160 27 41 3 12 4`
`160 27 41 3 12 99`
`1 999 999 1 99 999`
`999 999 1 1 99 9999999999999999`
	`160`
	`187`
	`3226`
	`999`
	`10000000000099799`

Note

For the fourth sample, the optimal solution is to hail 999 cabs, hailing a new one every meter. It seems quite weird, but you are confident that this is the optimal way to live the life in Night City.

This page is intentionally left blank.

Problem F. The Hermit

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

> *"Mine eyes doth see the Dao from the confines of my abode." Greetings, brave one!*
>
> — Xu Dog

The renowned alchemist Xu Dog discovered that precisely removing impurities could enhance the spiritual essence of the elixirs he was refining. Through day after day of alchemy, he found that the nature of these impurities was intricately related to mathematical problems. Since your progress in Dao is still shallow, Xu Dog decided to tell you the mathematical problem he needs to solve in the most straightforward way, rather than through the esoteric problems of alchemy.

Given two positive integers $n \le m$, calculate the sum of the answers to the following problem for all subsets of size n of $\{1, 2, \ldots, m\}$, modulo $998\,244\,353$:

- In a set of n numbers, you can remove some numbers so that the minimum value of the set is not equal to the greatest common divisor of the set. Find the maximum number of elements that can remain in the set after removal. If no non-empty subset satisfies the condition, the answer is defined as 0.

The greatest common divisor of a set is defined as the largest value among the common divisors of all elements in the set. For example, the greatest common divisor of the set $\{6, 9, 15\}$ is 3.

Input

Input consists of a single line containing two integers m, n ($1 \le n \le m \le 10^5$).

Output

Output an integer representing the answer, modulo $998\,244\,353$.

Examples

standard input	standard output
`4 3`	`7`
`11 4`	`1187`
`100000 99999`	`17356471`

Note

For the first example, all cases are listed below:

- $\{1, 2, 3\}$: $\{2, 3\}$
- $\{1, 2, 4\}$: No solution
- $\{1, 3, 4\}$: $\{3, 4\}$
- $\{2, 3, 4\}$: $\{2, 3, 4\}$

Therefore, the answer is $2 + 0 + 2 + 3 \bmod 998\,244\,353 = 7$.

This page is intentionally left blank.

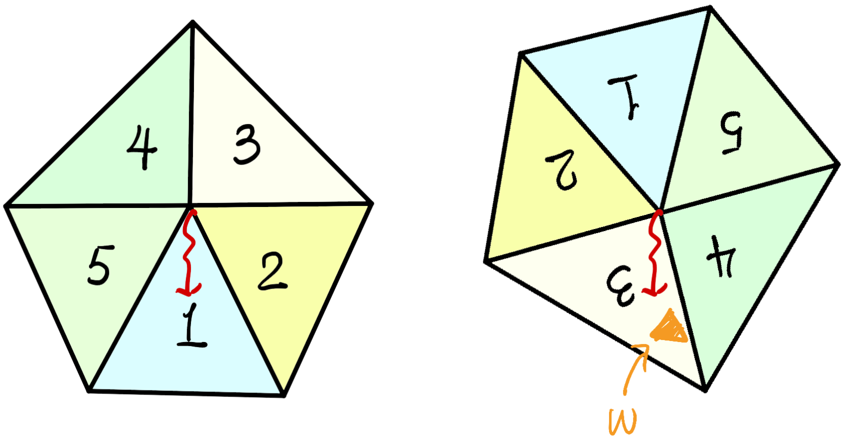
Problem G. The Wheel of Fortune

Input file:	`standard input`
Output file:	`standard output`
Time limit:	7 seconds
Memory limit:	1024 megabytes

The Hierophant and *The High Priestess* are fascinated by the mysteries of the universe. To explore some of these mysteries on Earth — apparently a part of the universe, they used a strange wheel to study the effect of gravity on random processes.

The wheel is a convex polygon, divided into several triangular regions by lines connecting the center of rotation to each vertex, with each triangular region representing a prize. After spinning the wheel, the prize is determined by the region directly below the stopping position. An unbiased wheel has its center of rotation at its centroid, making the final winning position dependent solely on the angle occupied by each region. Unfortunately, the wheel here may be biased: when the center of rotation is not at the centroid, the result of the wheel is **always** directed from the center of rotation to the centroid.

Clearly, a biased wheel is boring. To make various wheels meaningful, *The Hierophant* decided to perturb the result in the following way: placing a small magnetic weight at a **uniformly random** position within the wheel's area (considered as a point mass located inside the convex polygon) will change the centroid, thus affecting the result.



The first sample: the case without a magnet, and a possible case where the magnet changes the result.

The mass of the magnet is w, and the mass per unit area of the wheel is uniformly 1. The *High Priestess* wants to know the probability that each region becomes the final winning region after the aforementioned perturbation.

It can be proven that the probability measure for the wheel to uniquely determine the winning region (i.e., the center of rotation is at the centroid or the centroid is on the boundary of the region) is 0, so you can safely ignore such cases.

Input

The first line contains two integers n, w ($3 \le n \le 100000, 1 \le w \le 10^9$), representing the number of vertices of the wheel and the mass of the magnet.

The next n lines each contain two integers x_i, y_i ($|x_i|, |y_i| \le 30000$), representing the coordinates of the i-th vertex of the wheel, given in counterclockwise order along the boundary.

The last line contains two integers O_x, O_y ($|O_x|, |O_y| \le 30000$), representing the coordinates of the center of rotation, which is strictly inside the convex polygon.

The vertices of the wheel are all distinct, but three points may be collinear: it is guaranteed that when three adjacent points $u, v = (u \bmod n) + 1, w = (v \bmod n) + 1$ are collinear, point v is strictly on the segment connecting u and w, which ensures that all interior angles are within the range $(0, \pi]$.

It is guaranteed that the area S of the wheel satisfies: $\max\left\{\frac{S}{w}, \frac{w}{S}\right\} \le 1000$.

Output

Output n lines, each containing a real number p_i, representing the probability that the region bounded by the line connecting the i-th vertex and the $(i \bmod n) + 1$-th vertex is the final winning region. The answer will be considered correct if it has an absolute or relative error not exceeding 10^{-6}.

Examples

standard input	standard output

5 5
1 0
3 0
4 2
2 4
0 2
2 2
	0.313777778
	0.235555556
	0.107555556
	0.107555556
	0.235555556

8 8
0 0
1 0
2 0
2 1
2 2
1 2
0 2
0 1
1 1
	0.125000000
	0.125000000
	0.125000000
	0.125000000
	0.125000000
	0.125000000
	0.125000000
	0.125000000

3 3
-1 -10
1 -10
0 1
0 0
	1.000000000
	0.000000000
	0.000000000

4 36000000
-30000 -30000
30000 -30000
30000 30000
-30000 30000
1 0
	0.249998611
	0.248327778
	0.249998611
	0.251675000

4 2500
5 0
5 5
0 5
0 0
1 1
	0.402977500
	0.402977500
	0.097022500
	0.097022500

Problem H. Strength

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

People often use approximate methods to express numerical values in everyday conversations. Whether discussing time, money, or other numbers, people tend to round off to make communication simpler and easier. For example, if you and your friends dine at a restaurant with a bill of 98 yuan, many would say, "The bill is a hundred yuan," instead of using the exact number.

If we take a more aggressive approach and round multiple times, the final result can become absurd. For instance, you could round 145 up to 200, because 145 can be rounded to 150, which can then be rounded to 200; when someone says 2000, it could actually have been 2001, 1999, 1888, or even 11451 before rounding.

Given a number x, calculate the *uncertainty* of x within the range $[0, z]$, which is the count of numbers within the range $[0, z]$ that can be x after *aggressive rounding*. Here, *aggressive rounding* is defined as performing the following rounding operation arbitrarily (possibly zero) times:

- Let the decimal representation of x be $\overline{x_k x_{k-1} \dots x_1}$, and choose an index $i \in \{1, 2, \dots, k\}$.
- If $x_i < 5$, subtract $x_i \cdot 10^{i-1}$ from x;
- otherwise, add $(10 - x_i) \cdot 10^{i-1}$ to x.

Input

The input consists of multiple test cases. The first line contains a single integer T ($1 \le T \le 10^5$) — the number of test cases. The description of the test cases follows.

The input contains two integers in a line: x and z ($0 \le x, z \le 10^{18}$). Refer to above for their meanings.

Output

Each line contains a single number, representing the *uncertainty* of x in the range $[0, z]$.

Example

`standard input`	`standard output`
`5`	`2147483647`
`0 2147483646`	`55`
`10 100`	`0`
`671232353 1232363`	`1919810`
`123001006660996 3122507962333010`	`114514`
`100019990010301090 44519984489341188`	

Note

In the second test case, $10i + j$ ($i = 0, 1, \cdots, 9$, $j = 5, 6, 7, 8, 9$) and $10, 11, 12, 13, 14$ can be aggressively rounded to 10. Thus, the answer is 55.

This page is intentionally left blank.

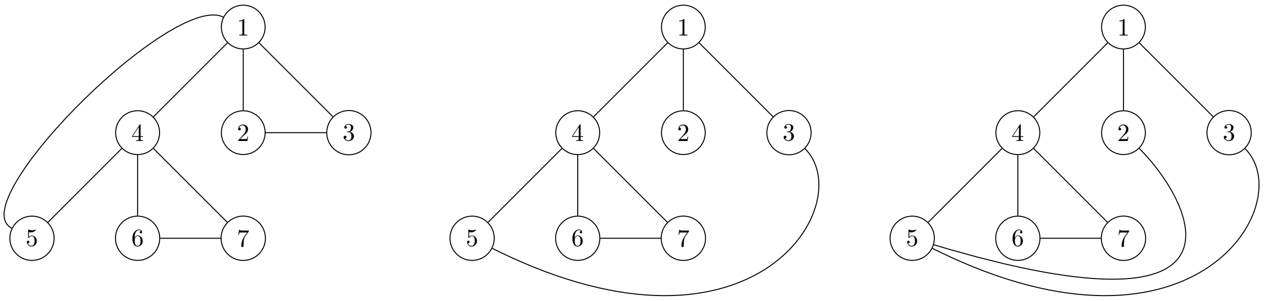
Problem I. The Hanged Man

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Claudette Morel is a botanist who is passionate about studying various plants. One day, while studying roses, she accidentally pricked her hand on a thorn. With her botany knowledge, she knew how to treat the wound, but more importantly, she wanted to prevent such incidents from happening again. To achieve this, she came up with a solution: make the rose thorns disappear.

The rose can be viewed as a tree with n nodes. To make the rose thornless, Claudette Morel can add several edges to the graph, as long as the addition does not create multiple edges or self-loops. However, she cannot add new nodes to the graph.

A simple graph is *thornless* if and only if each edge appears in exactly one simple cycle. A simple cycle is defined as a cycle that does not contain any repeated nodes (except for the starting and ending node being the same). The following illustrations explain what is a thornless graph and what is not.



- Left: This is a thornless graph.
- Middle: This is not a thornless graph because the edge $(1, 2)$ does not appear in any simple cycle.
- Right: This is not a thornless graph because the edge $(1, 2)$ appears in both cycles $1 \sim 2 \sim 5 \sim 4 \sim 1$ and $1 \sim 2 \sim 5 \sim 3 \sim 1$.

Now, Claudette Morel has taken out her roses, and you are tasked with analyzing whether they can be transformed into a thornless graph.

Input

The input consists of multiple test cases. The first line contains a single integer T $(1 \le T \le 10^5)$ — the number of test cases. The description of the test cases follows.

The first line contains one integer n $(2 \le n \le 3 \cdot 10^5)$ — the number of nodes in the tree.

Each of the following $n - 1$ lines contains two integers u_i and v_i $(1 \le u_i, v_i \le n)$, indicating that (u_i, v_i) is an edge on the tree.

It is guaranteed that the sum of n among T test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, if the tree cannot be transformed into a thornless graph, output -1.

Otherwise, on the first line, output k $(0 \le k \le n)$ — the number of edges you added.

In the following k lines, each line should contain two integers x_i and y_i $(1 \le x_i, y_i \le n)$ — the edges you added. Note that after adding edges, multiple edges and self-loops are not allowed. If there are multiple solutions, print any.

Example

standard input	standard output
3	-1
4	3
1 2	1 5
2 3	2 3
2 4	6 7
7	2
1 2	6 2
1 3	4 3
1 4	
4 5	
4 6	
4 7	
6	
1 2	
2 3	
2 4	
1 5	
5 6	

Note

The left graph in the statement shows the second test case.

Problem J. Temperance

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

> *In this world, only the green ones never let me down.*
>
> — Chen Loong

The renowned farmer Chen Loong discovered that a rational planting density could raise production.

Now, the farm can be viewed as a three-dimensional coordinate system, and a plant can be seen as a point in it. There are n different plants $A_i = (x_i, y_i, z_i)$. For each plant A_i, its *density* is defined as follows.

- Suppose there are a, b, and c plants other than A_i with the same x, y, or z coordinates as A_i, respectively. Then, the density of A_i is $\max\{a, b, c\}$.

Since Chen Loong's plants love involution, he decides to remove some plants with less density. Please answer the minimum number of plants that need to be removed such that each of the remaining plants has a density greater than or equal to k. Note that after removing a point, the density of other plants may change. In particular, removing all plants is always considered valid.

You need to solve for $k = 0, 1, \dots, n-1$ respectively.

Input

The input consists of multiple test cases. The first line contains a single integer T ($1 \le T \le 2 \times 10^4$) — the number of test cases. The description of the test cases follows.

The first line contains an integer n ($1 \le n \le 10^5$) — the number of plants.

In the next n lines, the i-th line contains three integers x_i, y_i, and z_i ($1 \le x_i, y_i, z_i \le 10^5$) — the coordinates of each plant.

It is guaranteed that the coordinates of the n plants are distinct.

It is guaranteed that the sum of n among T test cases does not exceed 2×10^5.

Output

For each test case, output n integers in a line, representing the answers for $k = 0, 1, \dots, n-1$.

Example

`standard input`	`standard output`

`2`
`5`
`1 1 1`
`1 1 2`
`1 1 3`
`2 3 5`
`2 2 4`
`3`
`1 1 1`
`2 2 2`
`3 3 3`
	`0 0 2 5 5`
	`0 3 3`

This page is intentionally left blank.

Problem K. The Devil

Input file:	`standard input`
Output file:	`standard output`
Time limit:	7 seconds
Memory limit:	1024 megabytes

People use abbreviations to express succinctly. This can lead to trouble when two phrases share the same abbreviation. For example, searching the term "`CCPC`" online, you will find not only the desired "`China Collegiate Programming Contest`", but also "`China Car Performance Challenge`", "`Competition and Consumer Protection Commission`", etc.

However, nobody will drive a car while writing code. You decide to resolve the problem by assigning a distinct initialism-like abbreviation to every phrase you often use. There are n phrases, each consisting of some words in lowercase and uppercase English letters. To create an abbreviation for a phrase, a non-empty prefix of each word in the phrase is chosen and then concatenated in order. For example, "`ChCoPrCo`" and "`CCPContest`" are valid abbreviations for "`China Collegiate Programming Contest`", but "`CCCP`" and "`CCPiC`" are not. Each phrase is abbreviated independently from the other phrases: the same word can be abbreviated differently in different phrases.

Construct an abbreviation scheme so that the total length of all abbreviations is minimized, while ensuring n abbreviations are distinct.

Input

The first line of input contains an integer n ($1 \le n \le 128$), representing the number of phrases.

The next n lines each contain a non-empty phrase consisting of no more than 128 non-empty words separated by single spaces. Each word consists of no more than 128 lowercase and uppercase English letters only. It is guaranteed that no two phrases are identical.

Output

If there is no possible solution, print "`no solution`" in a single line.

Otherwise, print n lines where the i-th line contains the abbreviation of the i-th phrase in the solution, in the order given by input. If there are multiple solutions, print any.

Examples

standard input	standard output

`5`
`automated teller machine`
`active teller machine`
`active trouble maker`
`always telling misinformation`
`American Teller Machinery`
	`atm`
	`atma`
	`actm`
	`atem`
	`ATM`

`5`
`Forest Conservation Committee Forum`
`Fuming Corruption Collusion Federation`
`Fulsome Cash Concealment Foundation`
`Funky Crony Capitalism Facilitator`
`Funny Cocky Cocky Funny`
	`FCCF`
	`FCoCF`
	`FuCCF`
	`FCCFa`
	`FCCFu`

`3`
`A AA`
`AA A`
`A A A`
	`no solution`

This page is intentionally left blank.

Problem L. The Tower

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Houraisan Kaguya is a NEET princess who loves to watch videos at home. Recently, she often watches videos on a famous video-sharing website called Mikufans.

There is a useful feature on Mikufans that allows users to leave a message during the video playback, which is called *danmaku*. Sometimes, there are so many *danmaku* messages at the same time that Kaguya cannot take them all in.

Source: `https://bilibili.com/video/BV1xx411c79H`

For simplicity, we only focus on the top *danmaku* messages: top *danmaku* messages are displayed at the top of the video screen, and each message occupies exactly one line. There is no limit to the number of *danmaku* messages at the same time (although Kaguya's screen will only display the first 10^9 lines, the remaining messages will still be correctly maintained in the overflow area of the screen).

During the video playback, there may be three types of events:

1. A **new** user sends some top *danmaku* messages. Each message will be placed at the topmost empty line in order.
2. The *danmaku* messages from a specific user disappear, and the lines they are in become empty lines. The other messages will **not** be affected and still remain in their positions.
3. Kaguya is interested in a *danmaku* message, so she wants to know the sender of the top *danmaku* message at a specific line.

Kaguya has many videos to watch every day, and she is too busy to re-watch the video from the beginning, so she asks you for help. Please help her find the senders of the *danmaku* messages.

Input

The first line of input contains one integer n ($1 \le n \le 5 \times 10^5$), representing the number of events.

Each of the following n lines contains one event in order. Each event is described in one of the following formats:

- `1 k`: A new user sends k ($1 \le k \le 10^9$) top *danmaku* messages. The ID of the user is the smallest positive integer that has not been used before.
- `2 u`: The *danmaku* messages from user u disappeared. It is guaranteed that the ID is valid, and the *danmaku* messages from user u have not disappeared before.
- `3 l`: Kaguya wants to know the ID of the sender of the *danmaku* message at the l-th ($1 \le l \le 10^9$) topmost line. If that line is empty, the answer is defined as `0`.

Output

For each query of type `3`, output the answer in a single line.

Examples

standard input	standard output

7
1 2
1 4
3 3
2 1
3 2
1 4
3 7
	2
	0
	3

5
3 6
3 8
1 2
1 5
3 2
	0
	0
	1

Problem M. Judgement

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

On a weekend, Qingshan and her friend Daniel created a one-player game called *Generalissimos*, in which players can draw great paintings.

The game is played on an $n \times m$ table. A cell has its color, which is one of red, blue, or white. Initially, cell (a, b) is red, cell (c, d) is blue (These two cells do not coincide), and others are white. We call (a, b) and (c, d) *special* cells and others *nonspecial*. During the game, the player can perform a certain operation, which consists of three steps:

1. The player selects a nonspecial cell (x, y).
2. The player selects another cell (x', y'). It must be non-white, and it must be neighboring to (x, y) (i.e., (x', y') and (x, y) must have a common edge).
3. The cell (x, y) is painted with the color of the cell (x', y').

In other words, in one operation, the player can color a nonspecial cell with the same color as its non-white neighboring cell. Note that a cell may be colored more than once, and the latest color will cover the earlier one.

The player can perform the operation any number of times and then stop the game. After that, the final configuration of the map table is printed.

Unfortunately, *Generalissimo* is full of cheats, and cheaters can color in any position at any time. In order to advocate *Justice*, you decide to write a judge program to determine whether the given configuration is possible to be a legal configuration in a normal game, or there must be a cheater.

Input

The input consists of multiple test cases. The first line contains a single integer T ($1 \le T \le 10^4$) — the number of test cases. The description of the test cases follows.

The first line contains two integers n, m ($1 \le n, m \le 500$ and $2 \le n \cdot m$) — the number of rows and columns.

The second line contains four integers a, b, c, and d ($1 \le a, c \le n$ and $1 \le b, d \le m$).

Each of the next n lines contains m characters. Each character is '`R`', '`B`', or '`.`', representing a red cell, a blue cell, and a white cell, respectively.

It is guaranteed that cell (a, b) and (c, d) do not coincide, and that the character on the a-th row b-th column and c-th row d-th column is '`R`' and '`B`', respectively.

It is guaranteed that the sum of $n \cdot m$ among T test cases does not exceed $250\,000$.

Output

For each test case, print "`YES`" (without quotes) if it is a legal configuration and "`NO`" (without quotes) otherwise.

You can print letters in any case (upper or lower).

Example

standard input	standard output
4 3 3 1 1 1 2 RBB RRR BBR 6 6 1 1 6 6 RRRRRR BBBBBR BRRRBR BRBBBR BRRRRR BBBBBB 5 5 3 3 4 4 BBR.B BBR.B RRR.B ...BB BBBB. 1 5 1 1 1 3 RBBBR	YES YES NO NO

Note

The following graph shows the first test case and how the player can reach the configuration without cheating. Each crown marks a special cell.