Problem A. Matrix

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Fill an $n \times n$ matrix with numbers in $[1, n^2]$, where each number occurs exactly once.

For a fixed number filling method, let a_i be the mininum number in the ith row, and $S = \{a_1, a_2, ..., a_n\} \cap \{1, 2, ..., n\}$.

You need to calculate $\sum |S| \pmod{998244353}$, i.e. the sum of the size of S over all possible methods.

Input

This problem contains multiple test cases.

The first line contains a single integer T $(1 \le T \le 30)$.

Then T cases follow, each of which contains a single interger n $(1 \le n \le 5000)$.

Output

For each test case, output one line contains the value of $\sum |S| \pmod{998244353}$.

Example

standard input	standard output
1 2	40

Problem B. Cypher

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

With the arrival of mechanical cryptography machines, cipher complexity exploded. A person enciphering or deciphering a message no longer needed to understand a cipher to use it, and messages which previously could take hours to encrypt were now enciphered almost instantly.

Lucida and Kanari were playing Counter-Strike: Global Offensive and needed frequent tactical exchanges. Losing interphone, they had to do radio communication, but the ease with which transmissions could be intercepted meant strong encryption became vital. They chose to adopt inventor Arthur Scherbius's Enigma machine, which was used by the German army during World War 2 and offered an unprecedented level of encryption.

When a letter key is pressed on the Enigma keyboard an electrical signal flows into a scrambler disc. This disc has 26 inputs and 26 outputs which are connected internally in a random fashion, so a signal entering at input 1 may exit at output 14. The signal passes through n of these discs then is reflected so it passes back through the same n discs again, before finally displaying the enciphered letter on a lampboard.

So far this is just a monoalphabetic substitution cipher, however, Enigma's strength comes from the fact that the discs rotate. The first disc rotates on every press, when it rotates a whole circle, it will cause the next disc to rotate and so the second will eventually cause the third disc to rotate and so on. This means the machine is a polyalphabetic cipher that can cycle through 26^n unique substitution alphabets before repeating.

To make the machine even more secure, up to 10 pairs of letters on the keyboard can be swapped using patch cables. Combined with the 26^n possible starting rotations for the scrambler discs the total number of initial configurations for the machine is large enough to avoid being deciphered.

As long as the machine is set to the correct initial setting it will decipher a message encrypted using the same settings, the initial setting is therefore the key. Lucida and Kanari used a new key for each game and sent out top-secret code books containing the initial Enigma settings every day. The Enigma worked well and they won most of the game by using the secret tactics.

Formally, the Enigma consists of three parts. They are combined with each other via signals. The signals indicate a location between 1 to 26, sometimes we may also use `A` to `Z` instead of 1 to 26 for convenience. When a key on the keyboard is pressed, there will be signals sent from the first part to the second part and then to the third part. After some process, signals will be sent back to the first part and finally become a letter shown on the lampboard.

The first part contains p patch cables that can swap p pairs of letters on the keyboard. It is guaranteed that the 2p letters are distinct. For example, if the letter `A` and `B` are swapped, after you pressed `A`, what's actually going into the second part is the signal of `B`(2); when a signal of `A`(1) comes from the second part, what's actually shown in front of you is letter `B`.

The second part are n discs which play the key role of the Enigma. Each disc can be seen as a monoalphabetic substitution cipher. Each disc has two rows up and down, and each row contains an `A` to `Z` permutation. Initially, letters on the first row are exactly `A` to `Z` and the disc can be rolled. The specific rules of disc rotation will be explained separately below. When the ith disc receives a signal(location j for example) from the $(i-1)$th disc(the signal to disc 1 is from the first part), it finds the letter at position j on the first row of itself, then it finds the position of the same letter on the second row and issues a signal to the $(i+1)$th disc. The signal of the nth disc is sent to the third part.

For example, at first, the first row of a disc is `ABCDEFGHIJKLMNOPQRSTUVWXYZ` and the second row is `ZYXWVUTSRQPONMLKJIHGFEDCBA`. If we roll the disc one time, the first row will become `BCDEFGHIJKLMNOPQRSTUVWXYZA` and the second row will become `YXWVUTSRQPONMLKJIHGFEDCBAZ`. If the disc receives a signal `C`(3) , because the third letter on the first row now is `D`, which is at the 22nd position of the second row. The disc should issue a signal 22 to the next disc.

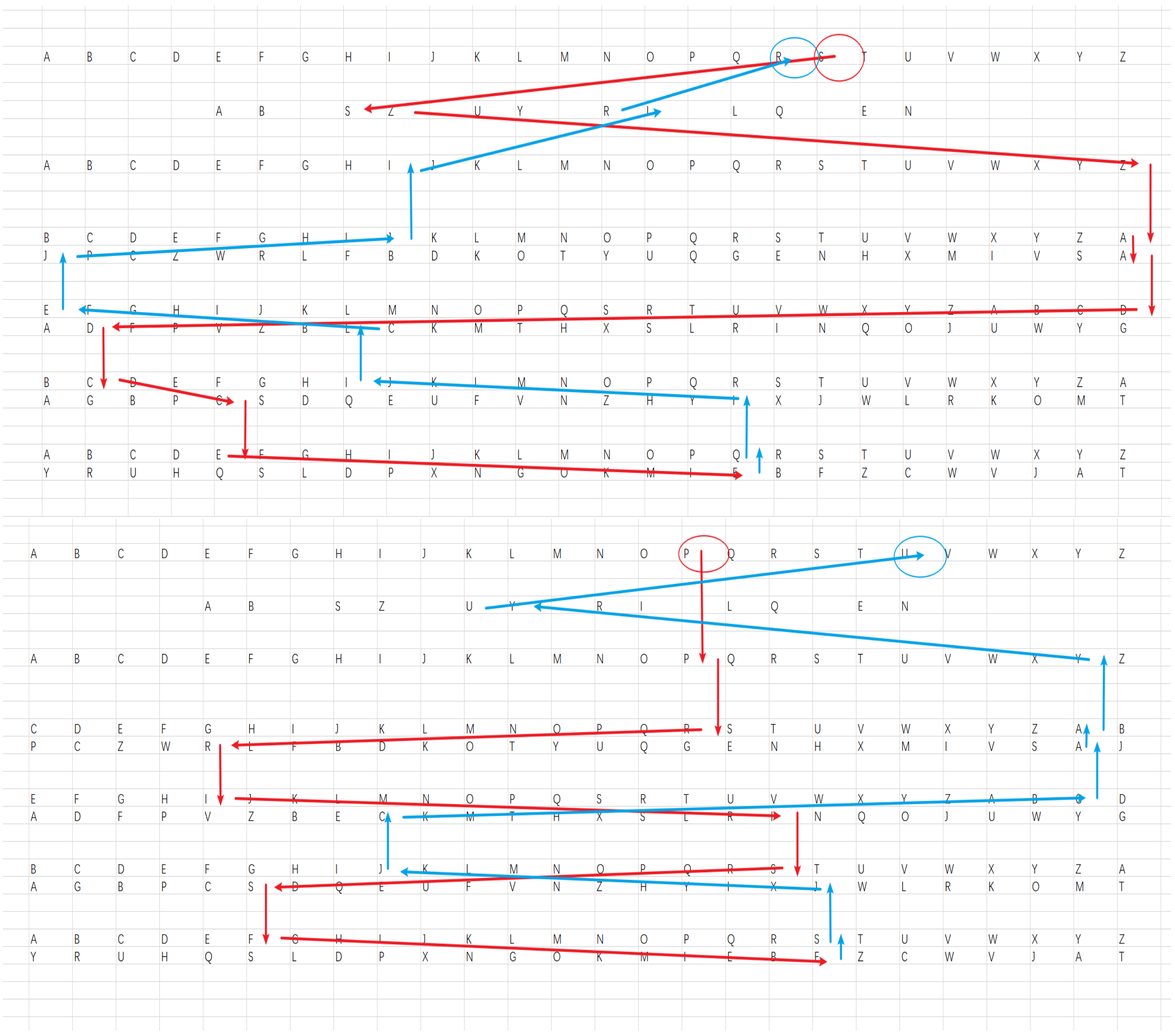
The third part, *reflector*, works as a function f from $\{\texttt{A}, \texttt{B}, ..., \texttt{Z}\}$ to $\{\texttt{A}, \texttt{B}, ..., \texttt{Z}\}$ where $f(x) \neq x$ and $f(f(x)) = x$. When the reflector receives a signal(location 5, letter `E` for example) from the nth disc of the second part, it issues a signal of $f(\texttt{E})$ back to the nth disc.

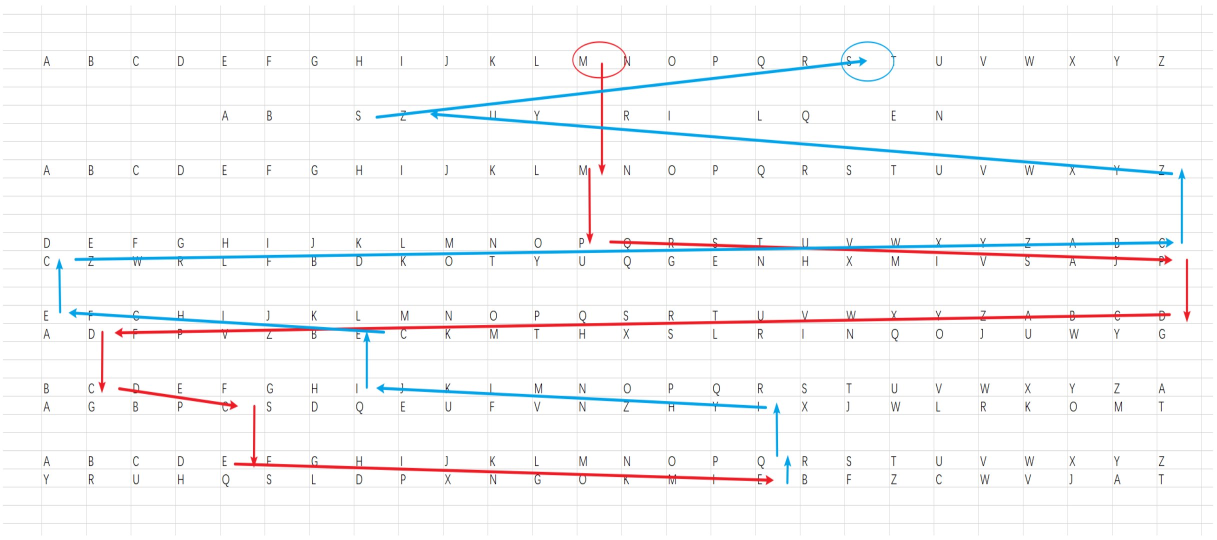
Then signals will be sent back to the 1st disc as it does from the 1st disc to the nth disc. The only difference is that this time each disc finds the letter at position j on the second row of itself and finds the position of the same letter on the first row.

Each time a key on the keyboard is pressed, the first disc rolls one time immediately (before any signal flow). Once the ith $(1 \leq i \leq n-1)$ disc is rolled T times $(T \geq 1, T \pmod{2}6 = 0)$, the $(i+1)$th disc will be rolled once. The rotation of the n-th disc has no effect.

Lucida and Kanari roll some discs at first and start to transmit the message. Now Lucida has sent Q messages to Kanari, please help Kanari to decipher these messages.

You can refer to the diagram below to better understand the usage of this Enigma. It shows the decryption process of the first three letters in the first sample.





Input

This problem contains multiple test cases.

The first line contains a single integer $T(1 \le T \le 25)$.

Then T test cases follow.

For each test case, on the first line is an integer p, the number of patch cables.$(1 \le p \le 10)$.

The next following p lines describe these patch cables. Each line contains two characters separated by no spaces, which means they are swapped by the patch cables.

Then an integer D on the next line, the number of discs. $(1 \le D \le 50)$

The next following D lines describe these discs. Each line contains a string of length 26. The string of the ith line is the second row of the ith disc initially. (The first row is `ABCDEFGHIJKLMNOPQRSTUVWXYZ` initially)

The next line contains D space-separated integers d_i, Lucida and Kanari roll the ith disc d_i times before they transmit. $(0 \le d_i \le 25)$.

The next line is a string r of length 26, which describes the reflector. $f(\mathtt{A}) = r_1, f(\mathtt{B}) = r_2, ..., f(\mathtt{Z}) = r_{26}$.

Then an integer Q on the next line, the number of messages Lucida sends to Kanari. $(1 \le Q \le 10$, and the length of each message is less than 50).

In the last Q lines, each line contains a message Lucida sends.

Output

For each message Lucida sends to Kanari, decipher it and output it in a single line. You need to output $\sum Q$ lines in total.

Example

standard input	standard output
3	RUSHB
6	C
AB	C
SZ	P
UY	C
RI	NORTHEAST
TO	QEM
EN	ZJVBU
3	CPUUG
AJPCZWRLFBDKOTYUQGENHXMIVS	QWK
UWYGADFPVZBECKMTHXSLRINQOJ	DAHJR
TAGBPCSDQEUFVNZHYIXJWLRKOM	WBKO
0 4 1	V
YRUHQSLDPXNGOKMIEBFZCWVJAT	GVEC
1	KJR
SPMFK	
3	
YE	
KV	
XA	
1	
LEPNXCURDYZWIQFKHGVSBAMJOT	
2	
WMVRLXOQTKJEBZGYHDUISCAFPN	
5	
O	
J	
B	
K	
ULQJMBQOM	
3	
JA	
BQ	
SG	
2	
MNZBFLVCGWTQOKEPYXSARHIJDU	
WXIFKOEVMQLZNUBDCSAPTRHGJY	
24 2	
KNEICYZODRATUBHQPJWLMXSVFG	
9	
THE	
QUCIK	
BROWN	
FOX	
JUMPS	
OVER	
A	
LAZT	
DOG	

Note

Though Lucida and Kanari didn't roll the first disc before they start, it was rolled for the first time as the moment Lucida pressed the first letter.

After a message was sent, Lucida and Kanari would not roll the discs back to their initial states.

In the third sample, when the second letter H is pressed, the first disc has already been rolled 26 times so it causes the second disc to roll.

In fact, when you input the plaintext, you will get the ciphertext. When you input the ciphertext into the cipher machine of the same state, you will also get the plaintext the same as your input before.

Problem C. Vertex Deletion

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	256 megabytes

Given a tree with n vertices, you can delete any vertex you like. We call a way of deletion beautiful if, in the remaining part of the tree, every vertex has at least one vertex still connected to it. You need to calculate how many ways of deletion are beautiful.

Two ways of deletion are considered the same if the set of the deleted vertex is the same.

Input

This problem contains multiple test cases.

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

For each test case, the first line contains one integer n ($1 \leq n \leq 10^5$).

The next $n-1$ lines each contains two integers u and v ($1 \leq u, v \leq n, u \neq v$), representing an edge (u, v).

It's guaranteed that $\sum n \leq 10^6$.

Output

Output T lines, each line contains an integer indicating the answer.

Since the answer can be very large, you only need to output the answer modulo 998244353.

Example

standard input	standard output

1
6
1 2
1 3
2 4
2 5
3 6
	22

Problem D. Lowbit

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

Lucida has a sequence of n integers $a_1, a_2, \ldots, a_n$. He asks you to perform two types of operations for him, which are described as follows:

1. `1 L R`, add $lowbit(a_i)$ to each a_i in the interval $[L, R]$.

2. `2 L R`, query the sum of the numbers in the interval $[L, R]$.

$lowbit(x)$ is defined as the largest power of 2 that divides x. For example, lowbit(4)=4, lowbit(5)=1. Specially, lowbit(0)=0.

Lucida wants you to give the answer modulo 998244353 for each of his queries.

Input

This problem contains multiple test cases.

The first line contains a single integer T $(1 \le T \le 20)$ indicating the number of test cases.

For each case, the first line contains an integer n $(1 \le n \le 10^5)$, the length of the sequence.

The next line contains n integers a_i $(1 \le a_i < 998244353)$ separated by spaces, representing the sequence.

The next line contains an integer m $(1 \le m \le 10^5)$, the number of operations.

The next m lines, each line contains 3 integers op, L, R $(1 \le op \le 2,\ 1 \le L \le R \le n)$, represents one operation. The specific meaning is described above.

Output

For each query, output a line contains the answer modulo 998244353.

Example

standard input	standard output

1
5
1 2 3 4 5
5
2 2 4
1 1 3
2 2 4
1 1 5
2 4 5
	9
	12
	14

Problem E. Easy Math Problem

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

Kanari recently worked on perfect numbers.

Perfect numbers are a special kind of natural numbers. A number is a perfect number if and only if the sum of all the true factors (divisors other than itself) is equal to itself. Such as $6 = 1 + 2 + 3$, $28 = 1 + 2 + 4 + 7 + 14$, etc.

Kanari made a kind of semi-perfect number of his own according to the definition of perfect number.

Let S be the set of all the true factors of the natural number X. If there is a subset of S such that the sum of the numbers in the subset is equal to the number itself, the number is said to be semi-perfect.

Obviously, all perfect numbers are semiperfect numbers. In addition, there are some numbers that are not perfect, and they also belong to Kanari's semi-perfect numbers. For example, the true factors set of 24 is $\{1, 2, 3, 4, 6, 8, 12\}$, we can select a subset $\{2, 4, 6, 12\}$, meet that $24 = 2 + 4 + 6 + 12$. So 24 can be called a semi-perfect number.

Kanari wants to know if he can find an integer k which is a multiple of positive integer p, such that k is a semi-perfect number.

Since Kanari is not good at math, he wants the k not to be too large ($k \leq 2 \times 10^{18}$), and the size of the subset is not larger than 1000. He wants you to give the subset you select.

Input

This problem contains multiple test cases.

The first line contains a single integer T ($1 \leq T \leq 4000$) indicating the number of test cases.

Then T cases follow, each of which contains a single interger p ($1 \leq p \leq 10^9$).

Output

Output $2T$ lines.

For each test case, if there is some integer k that satisfy the condition, output two space-separated integers on the first line, k ($k \leq 2 \times 10^{18}$) and n ($1 \leq n \leq 1000$) (the size of the subset you select). Then output n space-separated integers on the second line, the subset you select.

If you cannot find such k, output `-1` on the first line, and an empty line on the second line.

Please don't output extra space at the end of each line.

Example

standard input	standard output

2
6
1
	48 5
	2 4 6 12 24
	12 4
	1 2 3 6

Problem F. Permutation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	7 seconds
Memory limit:	256 megabytes

Given $\{a_i\}$, a permutation of $1...n$, you need to perform the following operations:

- Reverse a subarray indexed by $[L, R](1 \le L \le R \le |a|)$, that is, $\forall i \in [L, \frac{L+R}{2}]$, `swap(a[i],a[R+L-i])`.
- Complement the elements a_i such that $L \le a_i \le R(1 \le L \le R \le |a|)$, that is, $\forall L \le a_i \le R$, assign `R+L-a[i]` to `a[i]`.
- Increase all the elements that is no less than $v(1 \le v \le |a| + 1)$ by 1, and then insert v before the $i(1 \le i \le |a|)$th element. It's clear that the array becomes a permutation of $1...|a| + 1$ after that.
- Given $i(1 \le i \le |a|)$, print the value of a_i.
- Given $v(1 \le v \le |a|)$, print the position of v.

Input

This problem contains multiple test cases.

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

Each test case starts with a line of two integers $n, m(1 \le n, m \le 10^5)$, the initial length of the permutation and the number of operations.

Then follows a line of n numbers, representing a permutation of $1...n$.

For the following m lines, each line is of the following format:

- `1 L R`, asking you to reverse a subarray.
- `2 L R`, asking you to complement certain elements.
- `3 i v`, asking you to insert an element.
- `4 i`, asking you to print the value of the ith element.
- `5 v`, asking you to print the position of v.

Output

For each *print* operation, print a single integer in a line denoting the answer.

Example

standard input	standard output

1
5 5
1 2 3 4 5
1 1 3
2 1 5
3 1 2
4 6
5 2
	1
	1

Problem G. Ball

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

There are n grooves distributed on a slope evenly. We index the highest groove with n, while the lowest is 1. If we choose a number i, let a ball fall freely over the ith groove, three different situations may happen. If the ith groove is empty, then the ball will occupy it. If the ith groove is already occupied, the ball will roll down and occupy the first empty groove. If all the grooves below the ith groove are occupied, the ball will fall out of the slope.

Now Lucida has m balls. He wondered how many different ways for him to throw these balls such that there are exactly k of them falling out of the slope.

A way to throw these m balls can be represented by a sequence p($\{p_1, p_2, ..., p_n\}$, $1 \leq p_i \leq n$). p_i means that the ith ball is fall freely over the p_i groove.

Two ways are considered the same if the sequence p is the same.

Input

This problem contains multiple test cases.

The first line contains a single integer T ($1 \leq T \leq 1000$) indicating the number of test cases.

The following T lines each contains three integers n, m, k ($1 \leq n \leq 500$,$1 \leq m \leq 1000$, $0 \leq k \leq 500$, $k < m \leq n + k$), the number of grooves, the number of balls and the number of balls which fall out of the slope.

Output

For each test case, output one line contains one integer indicating the answer.

Since the answer can be very large, you only need to output the answer modulo 998244353.

Example

standard input	standard output
1 3 2 0	8

Note

For the sample, there are 3 grooves in total. We throw two balls and none of them falls out of the slope. There are 9 different ways to throw the balls and the only way that causes one of them to fall is to throw both of them over the first groove. So the answer will be $9 - 1 = 8$.

Problem H. Loneliness

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

The world that Kanari lives in is a grid of $n \times n$, and Kanari's home is on the top left corner $(1, 1)$. One day, he decided to take an adventure to the bottom right corner (n, n) of the world.

During the alone journey of Kanari, he feels lonely and he has 1 loneliness value at first. His sense of loneliness increases whenever he gets farther from home and decreases whenever he moves closer to it. To be specific, when he moves down, his loneliness value doubles; when he moves up, it halves; when he moves right, it is increased by 2 and when he moves left, it is decreased by 2.

More formally, let (x, y) be the current position of Kanari.

- If he moves down, he moves from (x, y) to $(x + 1, y)$
- If he moves up, he moves from (x, y) to $(x - 1, y)$
- If he moves left, he moves from (x, y) to $(x, y - 1)$
- If he moves right, he moves from (x, y) to $(x, y + 1)$

Kanari **can't move out of this world**. At any time, his coordinate (x, y) should satisfy that $1 \le x, y \le n$.

In addition, to better perceive his loneliness, Kanari hopes that his loneliness value will **always be a non-negative integer**. That is to say, if Kanari's loneliness value is odd, he will not be able to move up; if Kanari's loneliness value is 1 or 0, he will not be able to move left.

Kanari hopes that when he reaches the bottom right corner, his loneliness value is k. When Kanari reaches the destination but his loneliness value is not k, he should continue to move. However, if Kanari's loneliness value exceeds $2k$ **at any time**, he will go crazy because he is too lonely, so he needs to avoid that.

If it takes Kanari too much time to settle down, he will also go crazy, so he hopes that **he will not move more than 1000 times**.

He wonders if he can find such a route, so he asks you for help. The route can be represented as a string consisting of `U, D, L, R`. (U stands for up, D stands for down, L stands for left, R stands for right).

For example, if $n = 4$ and $k = 40$, Kanari can choose the route `RRDDLDRR`.

His position and his loneliness value are as follows:

No	Movement	Position	loneliness value
0	-	(1,1)	1
1	R	(1,2)	3
2	R	(1,3)	5
3	D	(2,3)	10
4	D	(3,3)	20
5	L	(3,2)	18
6	D	(4,2)	36
7	R	(4,3)	38
8	R	(4,4)	40

Pay attention: $n = 4$ **is just an example, the test data only contains the case of n** $=$ **100**.

Input

The first line contains two integers $T, n.(1 \le T \le 10^4, n = 100)$

Then T lines follow, each line contains a single integer k. $(1 \le k \le 10^{16})$

In the sample, just to show the format, $T = 1, n = 4, k = 40$, and your solution will not be tested using that sample. The first test is different from the first sample.

Output

For each test case, if there is no route for Kanari to get to the bottom right corner, output -1, otherwise, output a `UDLR` string with length less than 1000 to represent the route.

Example

standard input	standard output
1 4 40	RRDDLDRR

Problem I. Takeaway

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	256 megabytes

After CET-6, Kanari wants to order a takeaway to reward himself.

He opens *choutuan* app and starts to choose what to eat for his dinner. There are 7 types of dishes in total, the price of which are $7, 27, 41, 49, 63, 78, 108$ yuan respectively.

Kanari has three coupons:

1. When the price reaches 69 yuan, you will get 15 yuan off.
2. When the price reaches 89 yuan, you will get 30 yuan off.
3. When the price reaches 120 yuan, you will get 50 yuan off.

Kanari can only use one coupon each time and each coupon can only be used once. For each order, the app will always choose the best coupon for him.

For example, if the overall price of his order is 300 yuan, he will need to pay 250 yuan.

Now Kanari ordered n dishes, $a_1, a_2, a_3, ..., a_n$, denoting the types of the dishes respectively. He wants to know how much he will spend on his dinner.

Input

This problem contains multiple test cases.

The first line contains a single integer T $(1 \le T \le 10^6)$ indicating the number of test cases.

Then T cases follow, each of which contains two lines.

The first line of each test case contains an integer n $(1 \le n \le 7)$, the number of dishes Kanari ordered.

The second line contains n integers $a_1, a_2, ..., a_n$. $(1 \le a_i \le 7)$ indicating the dishes Kanari ordered.

Output

For each test case, output the money Kanari spends in a single line.

Example

standard input	standard output

`5`
`2`
`2 3`
`3`
`2 3 4`
`6`
`1 1 4 5 1 4`
`1`
`1`
`7`
`7 7 7 7 7 7 7`
	`68`
	`87`
	`132`
	`7`
	`706`

Problem J. Transform

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

Given you two points (A, B, C) and (x, y, z) in 3D space. Let L be the line cross the point (A, B, C) and the original point $(0, 0, 0)$. You will get two points P and Q if you rotate point (x, y, z) around line L by r degree and $-r$ degree. If the z coordinate of P is greater than Q, output P. Otherwise, output Q. We guarantee that the solution is unique.

Input

This problem contains multiple test cases.

The first line of the input contains an integer $T (1 \le T \le 50000)$, indicating the number of test cases.

Each of the following T lines contains seven integers A, B, C, x, y, z, r.

$(1 \le A, B, C, x, y, z \le 100, 1 \le r < 180)$.

Output

For each test case, output one line contains three real numbers indicates the coordinate of the answer.

Your answer will be accepted if the absolute or relative error between your answer and the standard answer is less than 10^{-6}.

Example

standard input	standard output

`1`
`1 2 3 4 5 6 7`
	`4.084934830 4.801379781 6.104101869`

Problem K. City

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Lucida occupies n cities connected by m undirected roads, and each road has a strength k_i. The enemy will attack to destroy these roads. When the enemy launches an attack with damage x, all roads with strength less than x will be destroyed.

Now Lucida has Q questions to ask you, how many pairs of cities are reachable to each other if the enemy launches an attack with damage p_i. City x and city y are reachable, which means that there is a path from x to y, and every road's strength in that path is greater than or equal to p_i.

Input

This problem contains multiple test cases.

The first line contains a single integer T $(1 \le T \le 10)$.

Then T test cases follow.

For each test case, the first line contains 3 integers n, m, Q $(2 \le n \le 10^5, 1 \le m \le 2\times10^5, 1 \le Q \le 2\times10^5)$, which represent the number of cities, the number of roads, and the number of queries.

The next m lines, each line contains three integers x, y, k $(1 \le x, y \le n,\ 1 \le k \le 10^9)$, which represent the road connecting city x and city y, and the strength of this road is k.

The next Q lines, each line contains an integer p_i $(1 \le p_i \le 10^9)$, asking how many pairs of cities are reachable to each other if the enemy launches an attack with damage p_i.

Output

Output $\sum_1^T Q$ lines, each line contains an integer, representing the answer for each question.

Example

standard input	standard output

1
5 5 3
1 2 2
2 3 3
3 4 1
4 5 1
5 1 3
3
2
1
	2
	6
	10

Problem L. kth Smallest Common Substring

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	256 megabytes

Given $n(1 \le n \le 10^5)$ strings containing only lowercase letters, you need to find out the kth smallest one among all the distinct common substrings of the n strings.

Input

This problem contains multiple test cases.

The first line contains an integer T $(1 \le T \le 20)$ indicating the number of test cases.

For each test case, the first line of the input contains an integer $n(1 \le n \le 10^5)$ indicates the number of the strings.

Each of the next n lines contains a string containing only lowercase letters.

The next line contains an integer q, the number of the queries.

Each of the next $q(1 \le q \le 10^5)$ lines contains an integer $k(1 \le k \le 10^7)$ represeting a query.

It is guaranteed that for each test case, the sum of the length of all strings is no more than 2×10^5.

Output

For each query, you need to give an interval $[l, r)$ indicating the first(leftmost) occurrence of the answer in the first string.

If the number of distinct common substrings is less than k, output -1 instead.

Example

standard input	standard output

1
5
abaaaa
aabaab
aabbba
aabaaa
abaabb
3
1
2
3
	0 1
	2 4
	0 2

Problem M. Master of Shuangpin

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

As you know, there are three kinds of Chinese input methods commonly used: Wubi, Pinyin and Shuangpin. With Shuangpin, you can type **any** Chinese word by pressing keys only twice.

Pinyin	Sequence	Pinyin	Sequence
q, iu	q	f, en	f
w, ei	w	g, eng	g
e	e	h, ang	h
r, uan	r	j, an	j
t, ue	t	k, uai, ing	k
y, un	y	l, uang, iang	l
u, sh	u	z, ou	z
i, ch	i	x, ia, ua	x
o, uo	o	c, ao	c
p, ie	p	v, zh, ui	v
a	a	b, in	b
s, ong, iong	s	n, iao	n
d, ai	d	m, ian	m

Attention that:

- For pinyin of length 1, you should repeat it in order to meet the conditions.
- For those of length 2, just output the original pinyin.
- For pinyin such as *ang*, you should press the first character of it and then look up this whole pinyin in the table for the second key.
- For simplification, **there is no character *v* in any input.** We believe that you, a Pinyin master, can tell *u* and *v* in any situations such as *lve* and *que*, so we **do not** challenge you here.

OK, now you are already a MASTER of Shuangpin! Please output the keys sequence to type the given sentences. For example, "ni hao shi jie"will be "ni hc ui jp".

Input

There are multiple test cases. Each line contains one.

Each line is a sequence of pinyin separated by spaces.

It is guaranteed that the number of test case is no more than 1000, the number of pinyin in one test case is no more than 500, and the number of pinyin in all the test cases is no more than 5000.

Output

The keys sequence separated by spaces.

Examples

standard input	standard output
rua ni xian qi po lan rang wo men dang qi shuang jiang cha na zhua zhu le wei lai zhe ti mian shen me wan yi	rx ni xm qi po lj rh wo mf dh qi ul jl ia na vx vu le ww ld ve ti mm uf me wj yi
ni you ben shi na lai mai a wo e le wo men chi shen me ang yang de dou zhi	ni yz bf ui na ld md aa wo ee le wo mf ii uf me ah yh de dz vi