Lista AA 2022 #6

A. Archaeology

1 second, 256 megabytes

Alice bought a Congo Prime Video subscription and was watching a documentary on the archaeological findings from Factor's Island on Loch Katrine in Scotland. The archaeologists found a book whose age and origin are unknown. Perhaps Alice can make some sense of it?

The book contains a single string of characters "a", "b" and "c". It has been pointed out that no two consecutive characters are the same. It has also been conjectured that the string contains an unusually long subsequence that reads the same from both sides.

Help Alice verify this by finding such subsequence that contains at least half of the characters of the original string, rounded down. Note that you don't have to maximise the length of it.

A string a is a subsequence of a string b if a can be obtained from b by deletion of several (possibly, zero or all) characters.

Input

The input consists of a single string s ($2 \leq |s| \leq 10^6$). The string s consists only of characters "a", "b", "c". It is guaranteed that no two consecutive characters are equal.

Output

Output a palindrome t that is a subsequence of s and $|t| \geq \lfloor \frac{|s|}{2} \rfloor$.

If there are multiple solutions, you may print any of them. You don't have to maximise the length of t.

If there are no solutions, output a string "IMPOSSIBLE" (quotes for clarity).

input
```
cacbac
```
output
```
aba
```

input
```
abc
```
output
```
a
```

input
```
cbacacacbcbababacbcb
```
output
```
cbaaacbcaaabc
```

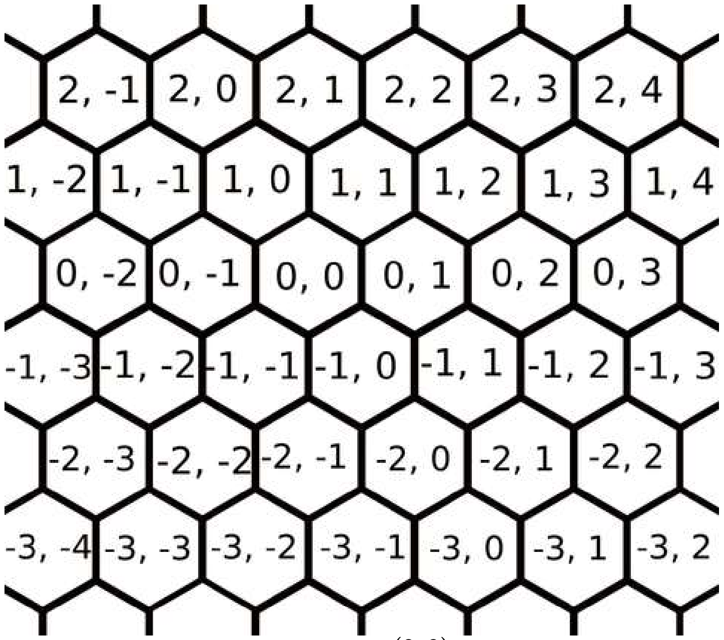
In the first example, other valid answers include "cacac", "caac", "aca" and "ccc".

B. Hexagons

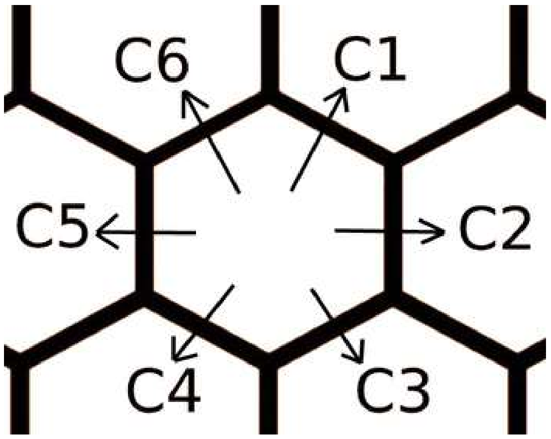
2 seconds, 256 megabytes

Lindsey Buckingham told Stevie Nicks "Go your own way". Nicks is now sad and wants to go away as quickly as possible, but she lives in a 2D hexagonal world.

Consider a hexagonal tiling of the plane as on the picture below.



Nicks wishes to go from the cell marked $(0, 0)$ to a certain cell given by the coordinates. She may go from a hexagon to any of its six neighbors you want, but there is a cost associated with each of them. The costs depend only on the direction in which you travel. Going from $(0, 0)$ to $(1, 1)$ will take the exact same cost as going from $(-2, -1)$ to $(-1, 0)$. The costs are given in the input in the order c_1, c_2, c_3, c_4, c_5, c_6 as in the picture below.



Print the smallest cost of a path from the origin which has coordinates $(0, 0)$ to the given cell.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^4$). Description of the test cases follows.

The first line of each test case contains two integers x and y ($-10^9 \leq x, y \leq 10^9$) representing the coordinates of the target hexagon.

The second line of each test case contains six integers c_1, c_2, c_3, c_4, c_5, c_6 ($1 \leq c_1, c_2, c_3, c_4, c_5, c_6 \leq 10^9$) representing the six costs of the making one step in a particular direction (refer to the picture above to see which edge is for each value).

Output

For each testcase output the smallest cost of a path from the origin to the given cell.

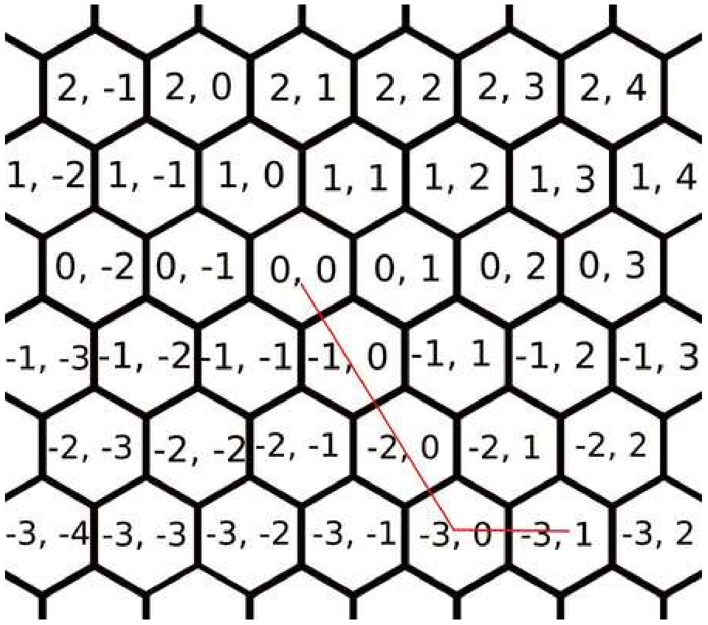
input

```
2
-3 1
1 3 5 7 9 11
1000000000 1000000000
1000000000 1000000000 1000000000 1000000000 1000000000 1000000000
```

output

```
18
1000000000000000000
```

The picture below shows the solution for the first sample. The cost 18 is reached by taking c_3 3 times and c_2 once, amounting to $5+5+5+3=18$.



C. GCD and MST

2 seconds, 256 megabytes

You are given an array a of n ($n \geq 2$) positive integers and an integer p. Consider an undirected weighted graph of n vertices numbered from 1 to n for which the edges between the vertices i and j ($i<j$) are added in the following manner:

- If $gcd(a_i, a_{i+1}, a_{i+2}, \dots, a_j) = min(a_i, a_{i+1}, a_{i+2}, \dots, a_j)$, then there is an edge of weight $min(a_i, a_{i+1}, a_{i+2}, \dots, a_j)$ between i and j.
- If $i+1=j$, then there is an edge of weight p between i and j.

Here $gcd(x, y, \dots)$ denotes the greatest common divisor (GCD) of integers x, y, ...

Note that there could be multiple edges between i and j if both of the above conditions are true, and if both the conditions fail for i and j, then there is no edge between these vertices.

The goal is to find the weight of the minimum spanning tree of this graph.

Input

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

The first line of each test case contains two integers n ($2 \leq n \leq 2 \cdot 10^5$) and p ($1 \leq p \leq 10^9$) — the number of nodes and the parameter p.

The second line contains n integers $a_1, a_2, a_3, \dots, a_n$ ($1 \leq a_i \leq 10^9$).

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

Output t lines. For each test case print the weight of the corresponding graph.

input

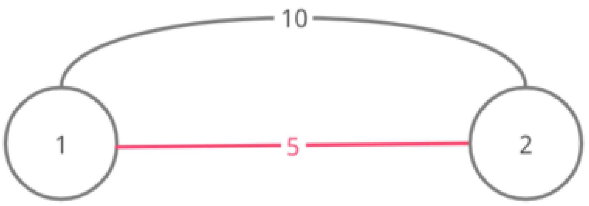
```
4
2 5
10 10
2 5
3 3
4 5
5 2 4 9
8 8
5 3 3 6 10 100 9 15
```

output

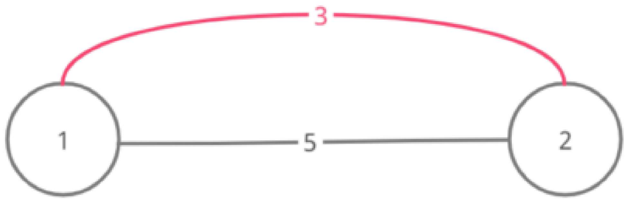
```
5
3
12
46
```

Here are the graphs for the four test cases of the example (the edges of a possible MST of the graphs are marked pink):

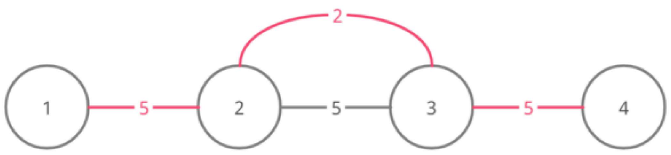
For test case 1



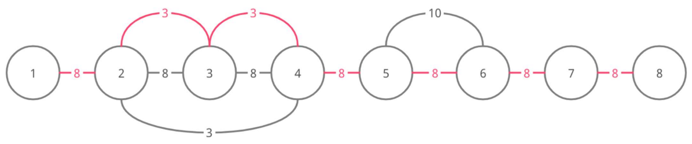
For test case 2



For test case 3



For test case 4



D. Dreamoon and Sets

1 second, 256 megabytes

Dreamoon likes to play with sets, integers and $\gcd$. $\gcd(a, b)$ is defined as the largest positive integer that divides both a and b.

Let S be a set of exactly four distinct integers greater than 0. Define S to be of rank k if and only if for all pairs of distinct elements s_i, s_j from S, $\gcd(s_i, s_j) = k$.

Given k and n, Dreamoon wants to make up n sets of rank k using integers from 1 to m such that no integer is used in two different sets (of course you can leave some integers without use). Calculate the minimum m that makes it possible and print one possible solution.

Input

The single line of the input contains two space separated integers n, k $(1 \le n \le 10\,000, 1 \le k \le 100)$.

Output

On the first line print a single integer — the minimal possible m.

On each of the next n lines print four space separated integers representing the i-th set.

Neither the order of the sets nor the order of integers within a set is important. If there are multiple possible solutions with minimal m, print any one of them.

input
```
1 1
```
output
```
5
1 2 3 5
```

input
```
2 2
```
output
```
22
2 4 6 22
14 18 10 16
```

For the first example it's easy to see that set $\{1, 2, 3, 4\}$ isn't a valid set of rank 1 since $\gcd(2, 4) = 2 \neq 1$.

E. Checkpoints

1 second, 512 megabytes

Gildong is developing a game consisting of n stages numbered from 1 to n. The player starts the game from the 1-st stage and should beat the stages in increasing order of the stage number. The player wins the game after beating the n-th stage.

There is at most one checkpoint on each stage, and there is always a checkpoint on the 1-st stage. At the beginning of the game, only the checkpoint on the 1-st stage is activated, and all other checkpoints are deactivated. When the player gets to the i-th stage that has a checkpoint, that checkpoint is activated.

For each try of a stage, the player can either beat the stage or fail the stage. If they beat the i-th stage, the player is moved to the $i+1$-st stage. If they fail the i-th stage, the player is moved to the most recent checkpoint they activated, and they have to beat the stages after that checkpoint again.

For example, assume that $n = 4$ and the checkpoints are on the 1-st and 3-rd stages. The player starts at the 1-st stage. If they fail on the 1-st stage, they need to retry the 1-st stage because the checkpoint on the 1-st stage is the most recent checkpoint they activated. If the player beats the 1-st stage, they're moved to the 2-nd stage. If they fail it, they're sent back to the 1-st stage again. If they beat both the 1-st stage and the 2-nd stage, they get to the 3-rd stage and the checkpoint on the 3-rd stage is activated. Now whenever they fail on the 3-rd stage, or the 4-th stage after beating the 3-rd stage, they're sent back to the 3-rd stage. If they beat both the 3-rd stage and the 4-th stage, they win the game.

Gildong is going to build the stages to have equal difficulty. He wants you to find any series of stages and checkpoints using at most 2000 stages, where the expected number of tries over all stages is exactly k, for a player whose probability of beating each stage is exactly $\frac{1}{2}$.

Input

Each test contains one or more test cases. The first line contains the number of test cases t $(1 \le t \le 50)$.

Each test case contains exactly one line. The line consists of a single integer k $(1 \le k \le 10^{18})$ — the expected number of tries over all stages Gildong wants to set for a player whose probability of beating each stage is exactly $\frac{1}{2}$.

Output

For each test case, print -1 if it's impossible to construct such a series of stages and checkpoints using at most 2000 stages.

Otherwise, print two lines. The first line should contain a single integer n ($1 \le n \le 2000$) – the number of stages. The second line should contain n integers, where the i-th integer represents whether the i-th stage has a checkpoint. The i-th integer should be 0 if the i-th stage doesn't have a checkpoint, and 1 if it has a checkpoint. Note that the first integer must be 1 according to the description.

input
```
4
1
2
8
12
```
output
```
-1
1
1
4
1 1 1 1
5
1 1 0 1 1
```

In the first and the second case, we can see that the 'easiest' series of stages is to have 1 stage with a checkpoint. This already requires 2 tries in expectation, so it is impossible to make it to require only 1 try.

In the third case, it takes 2 tries in expectation to beat each stage, and the player can always retry that stage without falling back to one of the previous stages if they fail it. Therefore the total expected number of tries is 8. Note that there exists an answer with fewer stages, but you are not required to minimize the number of stages used.

F. The Contest

2 seconds, 512 megabytes

A team of three programmers is going to play a contest. The contest consists of n problems, numbered from 1 to n. Each problem is printed on a separate sheet of paper. The participants have decided to divide the problem statements into three parts: the first programmer took some prefix of the statements (some number of first paper sheets), the third contestant took some suffix of the statements (some number of last paper sheets), and the second contestant took all remaining problems. But something went wrong — the statements were printed in the wrong order, so the contestants have received the problems in some random order.

The first contestant has received problems $a_{1,1}, a_{1,2}, \dots, a_{1,k_1}$. The second one has received problems $a_{2,1}, a_{2,2}, \dots, a_{2,k_2}$. The third one has received all remaining problems ($a_{3,1}, a_{3,2}, \dots, a_{3,k_3}$).

The contestants don't want to play the contest before they redistribute the statements. They want to redistribute them so that the first contestant receives some prefix of the problemset, the third contestant receives some suffix of the problemset, and the second contestant receives all the remaining problems.

During one move, some contestant may give one of their problems to other contestant. What is the minimum number of moves required to redistribute the problems?

It is possible that after redistribution some participant (or even two of them) will not have any problems.

Input

The first line contains three integers k_1, k_2 and k_3 ($1 \le k_1, k_2, k_3 \le 2 \cdot 10^5, k_1 + k_2 + k_3 \le 2 \cdot 10^5$) — the number of problems initially taken by the first, the second and the third participant, respectively.

The second line contains k_1 integers $a_{1,1}, a_{1,2}, \dots, a_{1,k_1}$ — the problems initially taken by the first participant.

The third line contains k_2 integers $a_{2,1}, a_{2,2}, \dots, a_{2,k_2}$ — the problems initially taken by the second participant.

The fourth line contains k_3 integers $a_{3,1}, a_{3,2}, \dots, a_{3,k_3}$ — the problems initially taken by the third participant.

It is guaranteed that no problem has been taken by two (or three) participants, and each integer $a_{i,j}$ meets the condition $1 \le a_{i,j} \le n$, where $n = k_1 + k_2 + k_3$.

Output

Print one integer — the minimum number of moves required to redistribute the problems so that the first participant gets the prefix of the problemset, the third participant gets the suffix of the problemset, and the second participant gets all of the remaining problems.

input
```
2 1 2
3 1
4
2 5
```
output
```
1
```

input
```
3 2 1
3 2 1
5 4
6
```
output
```
0
```

input
```
2 1 3
5 6
4
1 2 3
```
output
```
3
```

input
```
1 5 1
6
5 1 2 4 7
3
```
output
```
2
```

In the first example the third contestant should give the problem 2 to the first contestant, so the first contestant has 3 first problems, the third contestant has 1 last problem, and the second contestant has 1 remaining problem.

In the second example the distribution of problems is already valid: the first contestant has 3 first problems, the third contestant has 1 last problem, and the second contestant has 2 remaining problems.

The best course of action in the third example is to give all problems to the third contestant.

The best course of action in the fourth example is to give all problems to the second contestant.

G. Pairs

2 seconds, 256 megabytes

You have $2n$ integers $1, 2, \dots, 2n$. You have to redistribute these $2n$ elements into n pairs. After that, you choose x pairs and take minimum elements from them, and from the other $n - x$ pairs, you take maximum elements.

Your goal is to obtain the set of numbers $\{b_1, b_2, \dots, b_n\}$ as the result of taking elements from the pairs.

What is the number of different x-s ($0 \le x \le n$) such that it's possible to obtain the set b if for each x you can choose how to distribute numbers into pairs and from which x pairs choose minimum elements?

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case contains the integer n ($1 \le n \le 2 \cdot 10^5$).

The second line of each test case contains n integers $b_1, b_2, \dots, b_n$ ($1 \le b_1 < b_2 < \dots < b_n \le 2n$) — the set you'd like to get.

It's guaranteed that the sum of n over test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, print one number — the number of different x-s such that it's possible to obtain the set b.

input

```
3
1
1
5
1 4 5 9 10
2
3 4
```

output

```
1
3
1
```

In the first test case, $x = 1$ is the only option: you have one pair $(1, 2)$ and choose the minimum from this pair.

In the second test case, there are three possible x-s. If $x = 1$, then you can form the following pairs: $(1, 6)$, $(2, 4)$, $(3, 5)$, $(7, 9)$, $(8, 10)$. You can take minimum from $(1, 6)$ (equal to 1) and the maximum elements from all other pairs to get set b.

If $x = 2$, you can form pairs $(1, 2)$, $(3, 4)$, $(5, 6)$, $(7, 9)$, $(8, 10)$ and take the minimum elements from $(1, 2)$, $(5, 6)$ and the maximum elements from the other pairs.

If $x = 3$, you can form pairs $(1, 3)$, $(4, 6)$, $(5, 7)$, $(2, 9)$, $(8, 10)$ and take the minimum elements from $(1, 3)$, $(4, 6)$, $(5, 7)$.

In the third test case, $x = 0$ is the only option: you can form pairs $(1, 3)$, $(2, 4)$ and take the maximum elements from both of them.

H. Paths and Trees

3 seconds, 256 megabytes

Little girl Susie accidentally found her elder brother's notebook. She has many things to do, more important than solving problems, but she found this problem too interesting, so she wanted to know its solution and decided to ask you about it. So, the problem statement is as follows.

Let's assume that we are given a **connected** weighted undirected graph $G = (V, E)$ (here V is the set of vertices, E is the set of edges). The shortest-path tree from vertex u is such graph $G_1 = (V, E_1)$ that is a tree with the set of edges E_1 that is the subset of the set of edges of the initial graph E, and the lengths of the shortest paths from u to any vertex to G and to G_1 are the same.

You are given a **connected** weighted undirected graph G and vertex u. Your task is to find the shortest-path tree of the given graph from vertex u, the total weight of whose edges is minimum possible.

Input

The first line contains two numbers, n and m ($1 \le n \le 3 \cdot 10^5$, $0 \le m \le 3 \cdot 10^5$) — the number of vertices and edges of the graph, respectively.

Next m lines contain three integers each, representing an edge — u_i, v_i, w_i — the numbers of vertices connected by an edge and the weight of the edge ($u_i \ne v_i, 1 \le w_i \le 10^9$). It is guaranteed that graph is connected and that **there is no more than one edge between any pair of vertices**.

The last line of the input contains integer u ($1 \le u \le n$) — the number of the start vertex.

Output

In the first line print the minimum total weight of the edges of the tree.

In the next line print the indices of the edges that are included in the tree, separated by spaces. The edges are numbered starting from 1 in the order they follow in the input. You may print the numbers of the edges in any order.

If there are multiple answers, print any of them.

input

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

output

```
2
1 2
```

input

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

output

```
4
2 3 4
```

In the first sample there are two possible shortest path trees:

- with edges $1 – 3$ and $2 – 3$ (the total weight is 3);
- with edges $1 – 2$ and $2 – 3$ (the total weight is 2);

And, for example, a tree with edges $1 – 2$ and $1 – 3$ won't be a shortest path tree for vertex 3, because the distance from vertex 3 to vertex 2 in this tree equals 3, and in the original graph it is 1.

I. Frog Traveler

2 seconds, 512 megabytes

Frog Gorf is traveling through Swamp kingdom. Unfortunately, after a poor jump, he fell into a well of n meters depth. Now Gorf is on the bottom of the well and has a long way up.

The surface of the well's walls vary in quality: somewhere they are slippery, but somewhere have convenient ledges. In other words, if Gorf is on x meters below ground level, then in one jump he can go up on any integer distance from 0 to a_x meters inclusive. (Note that Gorf can't jump down, only up).

Unfortunately, Gorf has to take a break after each jump (including jump on 0 meters). And after jumping up to position x meters below ground level, he'll slip exactly b_x meters down while resting.

Calculate the minimum number of jumps Gorf needs to reach ground level.

Input

The first line contains a single integer n ($1 \le n \le 300\,000$) — the depth of the well.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le i$), where a_i is the maximum height Gorf can jump from i meters below ground level.

The third line contains n integers $b_1, b_2, \dots, b_n$ ($0 \le b_i \le n - i$), where b_i is the distance Gorf will slip down if he takes a break on i meters below ground level.

Output

If Gorf can't reach ground level, print -1. Otherwise, firstly print integer k — the minimum possible number of jumps.

Then print the sequence $d_1,\ d_2,\ \dots,\ d_k$ where d_j is the depth Gorf'll reach after the j-th jump, but before he'll slip down during the break. Ground level is equal to 0.

If there are multiple answers, print any of them.

input
```
3
0 2 2
1 1 0
```
output
```
2
1 0
```

input
```
2
1 1
1 0
```
output
```
-1
```

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

In the first example, Gorf is on the bottom of the well and jump to the height 1 meter below ground level. After that he slip down by meter and stays on height 2 meters below ground level. Now, from here, he can reach ground level in one jump.

In the second example, Gorf can jump to one meter below ground level, but will slip down back to the bottom of the well. That's why he can't reach ground level.

In the third example, Gorf can reach ground level only from the height 5 meters below the ground level. And Gorf can reach this height using a series of jumps $10 \Rightarrow 9 \dashrightarrow 9 \Rightarrow 4 \dashrightarrow 5$ where $\Rightarrow$ is the jump and $\dashrightarrow$ is slipping during breaks.

J. Bash and a Tough Math Puzzle

2.5 seconds, 256 megabytes

Bash likes playing with arrays. He has an array $a_1, a_2, \dots a_n$ of n integers. He likes to guess the greatest common divisor (gcd) of different segments of the array. Of course, sometimes the guess is not correct. However, Bash will be satisfied if his guess is *almost correct*.

Suppose he guesses that the gcd of the elements in the range $[l, r]$ of a is x. He considers the guess to be almost correct if he can change **at most** one element in the segment such that the gcd of the segment is x after making the change. Note that when he guesses, he doesn't actually change the array — he just wonders if the gcd of the segment can be made x. Apart from this, he also sometimes makes changes to the array itself.

Since he can't figure it out himself, Bash wants you to tell him which of his guesses are almost correct. Formally, you have to process q queries of one of the following forms:

- $1\ l\ r\ x$ — Bash guesses that the gcd of the range $[l, r]$ is x. Report if this guess is almost correct.
- $2\ i\ y$ — Bash sets a_i to y.

Note: The array is 1-indexed.

Input

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

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$) — the elements of the array.

The third line contains an integer q ($1 \le q \le 4 \cdot 10^5$) — the number of queries.

The next q lines describe the queries and may have one of the following forms:

- $1\ l\ r\ x$ ($1 \le l \le r \le n, 1 \le x \le 10^9$).
- $2\ i\ y$ ($1 \le i \le n, 1 \le y \le 10^9$).

Guaranteed, that there is at least one query of first type.

Output

For each query of first type, output "YES" (without quotes) if Bash's guess is almost correct and "NO" (without quotes) otherwise.

input
```
3
2 6 3
4
1 1 2 2
1 1 3 3
2 1 9
1 1 3 2
```
output
```
YES
YES
NO
```

input
```
5
1 2 3 4 5
6
1 1 4 2
2 3 6
1 1 4 2
1 1 5 2
2 5 10
1 1 5 2
```
output
```
NO
YES
NO
YES
```

In the first sample, the array initially is $\{2, 6, 3\}$.

For query 1, the first two numbers already have their gcd as 2.

For query 2, we can achieve a gcd of 3 by changing the first element of the array to 3. Note that the changes made during queries of type 1 are temporary and do not get reflected in the array.

After query 3, the array is now $\{9, 6, 3\}$.

For query 4, no matter which element you change, you cannot get the gcd of the range to be 2.