# Educational Codeforces Round 100 (Rated for Div. 2)

## A. Dungeon

2 seconds, 256 megabytes

You are playing a new computer game in which you have to fight monsters. In a dungeon you are trying to clear, you met three monsters; the first of them has $a$ health points, the second has $b$ health points, and the third has $c$.

To kill the monsters, you can use a cannon that, when fired, deals 1 damage to the selected monster. Every 7-th (i. e. shots with numbers 7, 14, 21 etc.) cannon shot is *enhanced* and deals 1 damage to **all** monsters, not just one of them. If some monster's current amount of health points is 0, it can't be targeted by a regular shot and does not receive damage from an *enhanced* shot.

You want to pass the dungeon beautifully, i. e., kill all the monsters with the same *enhanced* shot (i. e. after some *enhanced* shot, the health points of each of the monsters should become equal to 0 **for the first time**). Each shot must hit a monster, i. e. each shot deals damage to at least one monster.

### Input

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

Each test case consists of a single line that contains three integers $a$, $b$ and $c$ ($1 \le a, b, c \le 10^8$) — the number of health points each monster has.

### Output

For each test case, print `YES` if you can kill all the monsters with the same *enhanced* shot. Otherwise, print `NO`. You may print each letter in any case (for example, `YES`, `Yes`, `yes`, `yEs` will all be recognized as positive answer).

| input |
| --- |
| 3<br>3 2 4<br>1 1 1<br>10 1 7 |
| **output** |
| YES<br>NO<br>NO |

In the first test case, you can do as follows: 1-th shot to the first monster, 2-th shot to the second monster, 3-th shot to the third monster, 4-th shot to the first monster, 5-th shot to the third monster, 6-th shot to the third monster, and 7-th *enhanced* shot will kill all the monsters.

In the second test case, you can't kill monsters with the same *enhanced* shot, because the total number of health points of monsters is 3, and you will kill them in the first 3 shots.

## B. Find The Array

2 seconds, 256 megabytes

You are given an array $[a_1, a_2, \ldots, a_n]$ such that $1 \le a_i \le 10^9$. Let $S$ be the sum of all elements of the array $a$.

Let's call an array $b$ of $n$ integers **beautiful** if:

- $1 \le b_i \le 10^9$ for each $i$ from 1 to $n$;
- for every pair of adjacent integers from the array $(b_i, b_{i+1})$, either $b_i$ divides $b_{i+1}$, or $b_{i+1}$ divides $b_i$ (or both);
- $2 \sum_{i=1}^{n} |a_i - b_i| \le S$.

Your task is to find any beautiful array. It can be shown that at least one beautiful array always exists.

### Input

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

Each test case consists of two lines. The first line contains one integer $n$ ($2 \le n \le 50$).

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

### Output

For each test case, print the beautiful array $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10^9$) on a separate line. It can be shown that at least one beautiful array exists under these circumstances. If there are multiple answers, print any of them.

| input |
| --- |
| 4<br>5<br>1 2 3 4 5<br>2<br>4 6<br>2<br>1 1000000000<br>6<br>3 4 8 1 2 3 |
| **output** |
| 3 3 3 3 3<br>3 6<br>1 1000000000<br>4 4 8 1 3 3 |

## C. Busy Robot

2 seconds, 256 megabytes

You have a robot that can move along a number line. At time moment 0 it stands at point 0.

You give $n$ commands to the robot: at time $t_i$ seconds you command the robot to go to point $x_i$. Whenever the robot receives a command, it starts moving towards the point $x_i$ with the speed of 1 unit per second, and he stops when he reaches that point. However, while the robot is moving, it **ignores** all the other commands that you give him.

For example, suppose you give three commands to the robot: at time 1 move to point 5, at time 3 move to point 0 and at time 6 move to point 4. Then the robot stands at 0 until time 1, then starts moving towards 5, ignores the second command, reaches 5 at time 6 and immediately starts moving to 4 to execute the third command. At time 7 it reaches 4 and stops there.

You call the command $i$ successful, if there is a time moment in the range $[t_i, t_{i+1}]$ (i. e. after you give this command and before you give another one, both bounds inclusive; we consider $t_{n+1} = +\infty$) when the robot is at point $x_i$. Count the number of successful commands. Note that it is possible that an ignored command is successful.

### Input

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

The first line of a test case contains a single integer $n$ ($1 \le n \le 10^5$) — the number of commands.

The next $n$ lines describe the commands. The $i$-th of these lines contains two integers $t_i$ and $x_i$ ($1 \le t_i \le 10^9$, $-10^9 \le x_i \le 10^9$) — the time and the point of the $i$-th command.

The commands are ordered by time, that is, $t_i < t_{i+1}$ for all possible $i$.

The sum of $n$ over test cases does not exceed $10^5$.

**Output**
For each testcase output a single integer — the number of successful commands.

| input |
|---|
| 8<br>3<br>1 5<br>3 0<br>6 4<br>3<br>1 5<br>2 4<br>10 -5<br>5<br>2 -5<br>3 1<br>4 1<br>5 1<br>6 1<br>4<br>3 3<br>5 -3<br>9 2<br>12 0<br>8<br>1 1<br>2 -6<br>7 2<br>8 3<br>12 -9<br>14 2<br>18 -1<br>23 9<br>5<br>1 -4<br>4 -7<br>6 -1<br>7 -3<br>8 -7<br>2<br>1 2<br>2 -2<br>6<br>3 10<br>5 5<br>8 0<br>12 -4<br>14 -7<br>19 -5 |

| output |
|---|
| 1<br>2<br>0<br>2<br>1<br>1<br>0<br>2 |

The movements of the robot in the first test case are described in the problem statement. Only the last command is successful.

In the second test case the second command is successful: the robot passes through target point 4 at time 5. Also, the last command is eventually successful.

In the third test case no command is successful, and the robot stops at $-5$ at time moment 7.

Here are the 0-indexed sequences of the positions of the robot in each second for each testcase of the example. After the cut all the positions are equal to the last one:

1. [0, 0, 1, 2, 3, 4, 5, 4, 4, …]
2. [0, 0, 1, 2, 3, 4, 5, 5, 5, 5, 5, 4, 3, 2, 1, 0, −1, −2, −3, −4, −5, −5, …]
3. [0, 0, 0, −1, −2, −3, −4, −5, −5, …]
4. [0, 0, 0, 0, 1, 2, 3, 3, 3, 3, 2, 2, 2, 1, 0, 0, …]
5. [0, 0, 1, 0, −1, −2, −3, −4, −5, −6, −6, −6, −6, −7, −8, −9, −9, −9, −8, −7, −6, −5, −4, −3, −2, −1, −1, …]
6. [0, 0, −1, −2, −3, −4, −4, −3, −2, −1, −1, …]
7. [0, 0, 1, 2, 2, …]
8. [0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

## D. Pairs

2 seconds, 256 megabytes

You have $2n$ integers $1, 2, \ldots, 2n$. 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 $\{b_1, b_2, \ldots, b_n\}$ 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 $b_1, b_2, \ldots, b_n$ ($1 \le b_1 < b_2 < \cdots < b_n \le 2n$) — 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<br>1<br>1<br>5<br>1 4 5 9 10<br>2<br>3 4 |

| output |
|---|
| 1<br>3<br>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.

## E. Plan of Lectures

2 seconds, 512 megabytes

Ivan is a programming teacher. During the academic year, he plans to give $n$ lectures on $n$ different topics. Each topic should be used in exactly one lecture. Ivan wants to choose which topic will he explain during the 1-st, 2-nd, ..., $n$-th lecture — formally, he wants to choose some permutation of integers from 1 to $n$ (let's call this permutation $q$). $q_i$ is the index of the topic Ivan will explain during the $i$-th lecture.

For each topic (except **exactly one**), there exists a prerequisite topic (for the topic $i$, the prerequisite topic is $p_i$). Ivan cannot give a lecture on a topic before giving a lecture on its prerequisite topic. There exists at least one valid ordering of topics according to these prerequisite constraints.

Ordering the topics correctly can help students understand the lectures better. Ivan has $k$ special pairs of topics $(x_i, y_i)$ such that he knows that the students will understand the $y_i$-th topic better if the lecture on it is conducted **right after** the lecture on the $x_i$-th topic. Ivan wants to satisfy the constraints on every such pair, that is, for every $i \in [1, k]$, there should exist some $j \in [1, n - 1]$ such that $q_j = x_i$ and $q_{j+1} = y_i$.

Now Ivan wants to know if there exists an ordering of topics that satisfies all these constraints, and if at least one exists, find any of them.

### Input

The first line contains two integers $n$ and $k$ ($2 \le n \le 3 \cdot 10^5$, $1 \le k \le n - 1$) — the number of topics and the number of special pairs of topics, respectively.

The second line contains $n$ integers $p_1, p_2, \ldots, p_n$ ($0 \le p_i \le n$), where $p_i$ is the prerequisite topic for the topic $i$ (or $p_i = 0$ if the $i$-th topic has no prerequisite topics). Exactly one of these integers is $0$. At least one ordering of topics such that for every $i$ the $p_i$-th topic is placed before the $i$-th topic exists.

Then $k$ lines follow, the $i$-th line contains two integers $x_i$ and $y_i$ ($1 \le x_i, y_i \le n$; $x_i \ne y_i$) — the topics from the $i$-th special pair. All values of $x_i$ are pairwise distinct; similarly, all valus of $y_i$ are pairwise distinct.

### Output

If there is no ordering of topics meeting all the constraints, print $0$.

Otherwise, print $n$ pairwise distinct integers $q_1, q_2, \ldots, q_n$ ($1 \le q_i \le n$) — the ordering of topics meeting all of the constraints. If there are multiple answers, print any of them.

| input |
| --- |
| 5 2<br>2 3 0 5 3<br>1 5<br>5 4 |
| **output** |
| 3 2 1 5 4 |

| input |
| --- |
| 5 2<br>2 3 0 5 3<br>1 5<br>5 1 |
| **output** |
| 0 |

| input |
| --- |
| 5 1<br>2 3 0 5 3<br>4 5 |
| **output** |
| 0 |

| input |
| --- |
| 5 4<br>2 3 0 5 3<br>2 1<br>3 5<br>5 2<br>1 4 |
| **output** |
| 3 5 2 1 4 |

## F. Max Correct Set

4 seconds, 256 megabytes

Let's call the set of positive integers $S$ **correct** if the following two conditions are met:

- $S \subseteq \{1, 2, \ldots, n\}$;
- if $a \in S$ and $b \in S$, then $|a - b| \ne x$ and $|a - b| \ne y$.

For the given values $n$, $x$, and $y$, you have to find the maximum size of the **correct** set.

### Input

A single line contains three integers $n$, $x$ and $y$ ($1 \le n \le 10^9$; $1 \le x, y \le 22$).

### Output

Print one integer — the maximum size of the **correct** set.

| input |
| --- |
| 10 2 5 |
| **output** |
| 5 |

| input |
| --- |
| 21 4 6 |
| **output** |
| 9 |

| input |
| --- |
| 1337 7 7 |
| **output** |
| 672 |

| input |
| --- |
| 455678451 22 17 |
| **output** |
| 221997195 |