Codeforces Round #799 (Div. 4)

A. Marathon

1 second, 256 megabytes

You are given four **distinct** integers a, b, c, d.

Timur and three other people are running a marathon. The value a is the distance that Timur has run and b, c, d correspond to the distances the other three participants ran.

Output the number of participants in front of Timur.

Input

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

The description of each test case consists of four **distinct** integers a, b, c, d $(0 \le a, b, c, d \le 10^4)$.

Output

For each test case, output a single integer — the number of participants in front of Timur.

input
```
4
2 3 4 1
10000 0 1 2
500 600 400 300
0 9999 10000 9998
```
output
```
2
0
1
3
```

For the first test case, there are 2 people in front of Timur, specifically the participants who ran distances of 3 and 4. The other participant is not in front of Timur because he ran a shorter distance than Timur.

For the second test case, no one is in front of Timur, since he ran a distance of 10000 while all others ran a distance of 0, 1, and 2 respectively.

For the third test case, only the second person is in front of Timur, who ran a total distance of 600 while Timur ran a distance of 500.

B. All Distinct

1 second, 256 megabytes

Sho has an array a consisting of n integers. An operation consists of choosing two distinct indices i and j and removing a_i and a_j from the array.

For example, for the array $[2, 3, 4, 2, 5]$, Sho can choose to remove indices 1 and 3. After this operation, the array becomes $[3, 2, 5]$. Note that after any operation, the length of the array is reduced by two.

After he made some operations, Sho has an array that has only **distinct** elements. In addition, he made operations such that the resulting array is the **longest** possible.

More formally, the array after Sho has made his operations respects these criteria:

- No pairs such that ($i < j$) and $a_i = a_j$ exist.
- The length of a is maximized.

Output the length of the final array.

Input

The first line contains a single integer t $(1 \le t \le 10^3)$ — the number of test cases.

The first line of each test case contains a single integer n ($1 \le n \le 50$) — the length of the array.

The second line of each test case contains n integers a_i ($1 \le a_i \le 10^4$) — the elements of the array.

Output

For each test case, output a single integer — the length of the final array. Remember that in the final array, all elements are different, and its length is maximum.

input
```
4
6
2 2 2 3 3 3
5
9 1 9 9 1
4
15 16 16 15
4
10 100 1000 10000
```
output
```
2
1
2
4
```

For the first test case Sho can perform operations as follows:

1. Choose indices 1 and 5 to remove. The array becomes $[2, 2, 2, 3, 3, 3] \rightarrow [2, 2, 3, 3]$.
2. Choose indices 1 and 4 to remove. The array becomes $[2, 2, 3, 3] \rightarrow [2, 3]$.

The final array has a length of 2, so the answer is 2. It can be proven that Sho cannot obtain an array with a longer length.
For the second test case Sho can perform operations as follows:

1. Choose indices 3 and 4 to remove. The array becomes $[9, 1, 9, 9, 1] \rightarrow [9, 1, 1]$.
2. Choose indices 1 and 3 to remove. The array becomes $[9, 1, 1] \rightarrow [1]$.

The final array has a length of 1, so the answer is 1. It can be proven that Sho cannot obtain an array with a longer length.

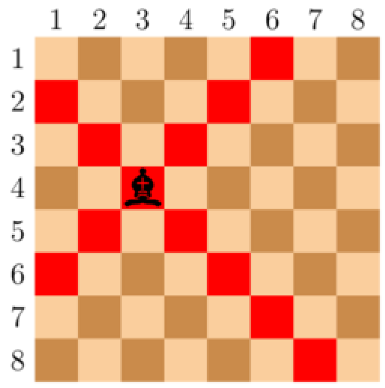
C. Where's the Bishop?

1 second, 256 megabytes

Mihai has an 8×8 chessboard whose rows are numbered from 1 to 8 from top to bottom and whose columns are numbered from 1 to 8 from left to right.

Mihai has placed exactly one bishop on the chessboard. **The bishop is not placed on the edges of the board.** (In other words, the row and column of the bishop are between 2 and 7, inclusive.)

The bishop attacks in all directions diagonally, and there is no limit to the distance which the bishop can attack. Note that the cell on which the bishop is placed is also considered attacked.



An example of a bishop on a chessboard. The squares it attacks are marked in red.

Mihai has marked all squares the bishop attacks, but forgot where the bishop was! Help Mihai find the position of the bishop.

Input

The first line of the input contains a single integer t $(1 \leq t \leq 36)$ — the number of test cases. The description of test cases follows. There is an empty line before each test case.

Each test case consists of 8 lines, each containing 8 characters. Each of these characters is either '`#`' or '`.`', denoting a square under attack and a square not under attack, respectively.

Output

For each test case, output two integers r and c $(2 \leq r, c \leq 7)$ — the row and column of the bishop.

The input is generated in such a way that there is always exactly one possible location of the bishop that is not on the edge of the board.

input

```
3

.....#..
#...#...
.#.#....
..#.....
.#.#....
#...#...
.....#..
......#.

#.#.....
.#......
#.#.....
...#....
....#...
.....#..
......#.
.......#

.#.....#
..#...#.
...#.#..
....#...
...#.#..
..#...#.
.#.....#
#.......
```

output

```
4 3
2 2
4 5
```

The first test case is pictured in the statement. Since the bishop lies in the intersection row 4 and column 3, the correct output is `4 3`.

D. The Clock

1 second, 256 megabytes

Victor has a 24-hour clock that shows the time in the format "`HH:MM`" ($00 \leq \texttt{HH} \leq 23$, $00 \leq \texttt{MM} \leq 59$). He looks at the clock every x minutes, and the clock is currently showing time s.

How many **different** palindromes will Victor see in total after looking at the clock every x minutes, the first time being at time s?

For example, if the clock starts out as `03:12` and Victor looks at the clock every 360 minutes (i.e. every 6 hours), then he will see the times `03:12`, `09:12`, `15:12`, `21:12`, `03:12`, and the times will continue to repeat. Here the time `21:12` is the only palindrome he will ever see, so the answer is 1.

A palindrome is a string that reads the same backward as forward. For example, the times `12:21`, `05:50`, `11:11` are palindromes but `13:13`, `22:10`, `02:22` are not.

Input

The first line of the input contains an integer t $(1 \leq t \leq 100)$ — the number of test cases. The description of each test case follows.

The only line of each test case contains a string s of length 5 with the format "`HH:MM`" where "`HH`" is from "`00`" to "`23`" and "`MM`" is from "`00`" to "`59`" (both "`HH`" and "`MM`" have exactly two digits) and an integer x ($1 \leq x \leq 1440$) — the number of minutes Victor takes to look again at the clock.

Output

For each test case, output a single integer — the number of different palindromes Victor will see if he looks at the clock every x minutes starting from time s.

input

```
6
03:12 360
00:00 1
13:22 2
15:15 10
11:11 1440
22:30 27
```

output

```
1
16
10
0
1
1
```

The first test case is explained in the statement.

E. Binary Deque

2 seconds, 256 megabytes

Slavic has an array of length n consisting only of zeroes and ones. In one operation, he removes either the first or the last element of the array.

What is the minimum number of operations Slavic has to perform such that the total sum of the array is equal to s after performing all the operations? In case the sum s can't be obtained after any amount of operations, you should output `-1`.

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 and s ($1 \leq n, s \leq 2 \cdot 10^5$) — the length of the array and the needed sum of elements.

The second line of each test case contains n integers a_i $(0 \leq a_i \leq 1)$ — the elements of the array.

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

Output

For each test case, output a single integer — the minimum amount of operations required to have the total sum of the array equal to s, or `-1` if obtaining an array with sum s isn't possible.

input

```
7
3 1
1 0 0
3 1
1 1 0
9 3
0 1 0 1 1 1 0 0 1
6 4
1 1 1 1 1 1
5 1
0 0 1 1 0
16 2
1 1 0 0 1 0 0 1 1 0 0 0 0 0 1 1
6 3
1 0 1 0 0 0
```

output

```
0
1
3
2
2
7
-1
```

In the first test case, the sum of the whole array is 1 from the beginning, so we don't have to make any operations.

In the second test case, the sum of the array is 2 and we want it to be equal to 1, so we should remove the first element. The array turns into $[1, 0]$, which has a sum equal to 1.

In the third test case, the sum of the array is 5 and we need it to be 3. We can obtain such a sum by removing the first two elements and the last element, doing a total of three operations. The array turns into $[0, 1, 1, 1, 0, 0]$, which has a sum equal to 3.

F. 3SUM

1 second, 256 megabytes

Given an array a of positive integers with length n, determine if there exist three **distinct** indices i, j, k such that $a_i + a_j + a_k$ ends in the digit 3.

Input

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

The first line of each test case contains an integer n ($3 \leq n \leq 2 \cdot 10^5$) — the length of the array.

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

The sum of n across all test cases does not exceed $2 \cdot 10^5$.

Output

Output t lines, each of which contains the answer to the corresponding test case. Output "YES" if there exist three **distinct** indices i, j, k satisfying the constraints in the statement, and "NO" otherwise.

You can output the answer in any case (for example, the strings "yEs", "yes", "Yes" and "YES" will be recognized as a positive answer).

input

```
6
4
20 22 19 84
4
1 11 1 2022
4
1100 1100 1100 1111
5
12 34 56 78 90
4
1 9 8 4
6
16 38 94 25 18 99
```

output

```
YES
YES
NO
NO
YES
YES
```

In the first test case, you can select $i = 1$, $j = 4$, $k = 3$. Then $a_1 + a_4 + a_3 = 20 + 84 + 19 = 123$, which ends in the digit 3.

In the second test case, you can select $i = 1$, $j = 2$, $k = 3$. Then $a_1 + a_2 + a_3 = 1 + 11 + 1 = 13$, which ends in the digit 3.

In the third test case, it can be proven that no such i, j, k exist. Note that $i = 4$, $j = 4$, $k = 4$ is **not** a valid solution, since although $a_4 + a_4 + a_4 = 1111 + 1111 + 1111 = 3333$, which ends in the digit 3, the indices need to be **distinct**.

In the fourth test case, it can be proven that no such i, j, k exist.

In the fifth test case, you can select $i = 4$, $j = 3$, $k = 1$. Then $a_4 + a_3 + a_1 = 4 + 8 + 1 = 13$, which ends in the digit 3.

In the sixth test case, you can select $i = 1$, $j = 2$, $k = 6$. Then $a_1 + a_2 + a_6 = 16 + 38 + 99 = 153$, which ends in the digit 3.

G. 2^Sort

1 second, 256 megabytes

Given an array a of length n and an integer k, find the number of indices $1 \leq i \leq n - k$ such that the subarray $[a_i, \dots, a_{i+k}]$ with length $k + 1$ (**not** with length k) has the following property:

- If you multiply the first element by 2^0, the second element by 2^1, ..., and the ($k + 1$)-st element by 2^k, then this subarray is sorted in strictly increasing order.

More formally, count the number of indices $1 \leq i \leq n - k$ such that

$$2^0 \cdot a_i < 2^1 \cdot a_{i+1} < 2^2 \cdot a_{i+2} < \dots < 2^k \cdot a_{i+k}.$$

Input

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

The first line of each test case contains two integers n, k ($3 \leq n \leq 2 \cdot 10^5$, $1 \leq k < n$) — the length of the array and the number of inequalities.

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

The sum of n across all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer — the number of indices satisfying the condition in the statement.

input

```
6
4 2
20 22 19 84
5 1
9 5 3 2 1
5 2
9 5 3 2 1
7 2
22 12 16 4 3 22 12
7 3
22 12 16 4 3 22 12
9 3
3 9 12 3 9 12 3 9 12
```

output

```
2
3
2
3
1
0
```

In the first test case, both subarrays satisfy the condition:

- $i = 1$: the subarray $[a_1, a_2, a_3] = [20, 22, 19]$, and $1 \cdot 20 < 2 \cdot 22 < 4 \cdot 19$.
- $i = 2$: the subarray $[a_2, a_3, a_4] = [22, 19, 84]$, and $1 \cdot 22 < 2 \cdot 19 < 4 \cdot 84$.

In the second test case, three subarrays satisfy the condition:

- $i = 1$: the subarray $[a_1, a_2] = [9, 5]$, and $1 \cdot 9 < 2 \cdot 5$.
- $i = 2$: the subarray $[a_2, a_3] = [5, 3]$, and $1 \cdot 5 < 2 \cdot 3$.
- $i = 3$: the subarray $[a_3, a_4] = [3, 2]$, and $1 \cdot 3 < 2 \cdot 2$.
- $i = 4$: the subarray $[a_4, a_5] = [2, 1]$, but $1 \cdot 2 = 2 \cdot 1$, so this subarray doesn't satisfy the condition.

H. Gambling

2 seconds, 256 megabytes

Marian is at a casino. The game at the casino works like this.

Before each round, the player selects a number between 1 and 10^9. After that, a dice with 10^9 faces is rolled so that a random number between 1 and 10^9 appears. If the player guesses the number correctly their total money is doubled, else their total money is halved.

Marian predicted the future and knows all the numbers $x_1, x_2, \dots, x_n$ that the dice will show in the next n rounds.

He will pick three integers a, l and r ($l \leq r$). He will play $r - l + 1$ rounds (rounds between l and r inclusive). In each of these rounds, he will guess the same number a. At the start (before the round l) he has 1 dollar.

Marian asks you to determine the integers a, l and r ($1 \leq a \leq 10^9$, $1 \leq l \leq r \leq n$) such that he makes the most money at the end.

Note that during halving and multiplying there is no rounding and there are no precision errors. So, for example during a game, Marian could have money equal to $\frac{1}{1024}$, $\frac{1}{128}$, $\frac{1}{2}$, 1, 2, 4, etc. (any value of 2^t, where t is an integer of any sign).

Input

The first line contains a single integer t ($1 \leq t \leq 100$) — the number of test cases.

The first line of each test case contains a single integer n ($1 \leq n \leq 2 \cdot 10^5$) — the number of rounds.

The second line of each test case contains n integers $x_1, x_2, \dots, x_n$ ($1 \leq x_i \leq 10^9$), where x_i is the number that will fall on the dice in the i-th round.

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

Output

For each test case, output three integers a, l, and r such that Marian makes the most amount of money gambling with his strategy. If there are multiple answers, you may output any of them.

input

```
4
5
4 4 3 4 4
5
11 1 11 1 11
1
1000000000
10
8 8 8 9 9 6 6 9 6 6
```

output

```
4 1 5
1 2 2
1000000000 1 1
6 6 10
```

For the first test case, the best choice is $a = 4$, $l = 1$, $r = 5$, and the game would go as follows.

- Marian starts with one dollar.
- After the first round, he ends up with 2 dollars because the numbers coincide with the chosen one.
- After the second round, he ends up with 4 dollars because the numbers coincide again.
- After the third round, he ends up with 2 dollars because he guesses 4 even though 3 is the correct choice.
- After the fourth round, he ends up with 4 dollars again.
- In the final round, he ends up 8 dollars because he again guessed correctly.

There are many possible answers for the second test case, but it can be proven that Marian will not end up with more than 2 dollars, so any choice with $l = r$ with the appropriate a is acceptable.