Codeforces Round #748 (Div. 3)

A. Elections

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

The elections in which three candidates participated have recently ended. The first candidate received a votes, the second one received b votes, the third one received c votes. For each candidate, solve the following problem: how many votes should be added to this candidate so that he wins the election (i.e. the number of votes for this candidate was strictly greater than the number of votes for any other candidate)?

Please note that for each candidate it is necessary to solve this problem **independently**, i.e. the added votes for any candidate **do not** affect the calculations when getting the answer for the other two candidates.

Input

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

Each test case consists of one line containing three integers a, b, and c ($0 \le a, b, c \le 10^9$).

Output

For each test case, output in a separate line three integers A, B, and C ($A, B, C \ge 0$) separated by spaces — the answers to the problem for the first, second, and third candidate, respectively.

Example

input
5 0 0 0 10 75 15 13 13 17 1000 0 0 0 1000000000 0
output
1 1 1 66 0 61 5 5 0 0 1001 1001 1000000001 0 1000000001

B. Make it Divisible by 25

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

It is given a positive integer n. In 1 move, one can select any single digit and remove it (i.e. one selects some position in the number and removes the digit located at this position). The operation cannot be performed if only one digit remains. If the resulting number contains leading zeroes, they are automatically removed.

E.g. if one removes from the number 32925 the 3-rd digit, the resulting number will be 3225. If one removes from the number 20099050 the first digit, the resulting number will be 99050 (the 2 zeroes going next to the first digit are automatically removed).

What is the minimum number of steps to get a number such that it is divisible by 25 and **positive**? It is guaranteed that, for each n occurring in the input, the answer exists. It is guaranteed that the number n has no leading zeros.

Input

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

Each test case consists of one line containing one integer n ($25 \le n \le 10^{18}$). It is guaranteed that, for each n occurring in the input, the answer exists. It is guaranteed that the number n has no leading zeros.

Output

For each test case output on a separate line an integer k ($k \ge 0$) — the minimum number of steps to get a number such that it is divisible by 25 and positive.

Example

input

```
5
100
71345
3259
50555
2050047
```

output

```
0
3
1
3
2
```

Note

In the first test case, it is already given a number divisible by 25.

In the second test case, we can remove the digits 1, 3, and 4 to get the number 75.

In the third test case, it's enough to remove the last digit to get the number 325.

In the fourth test case, we can remove the three last digits to get the number 50.

In the fifth test case, it's enough to remove the digits 4 and 7.

C. Save More Mice

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are one cat, k mice, and one hole on a coordinate line. The cat is located at the point 0, the hole is located at the point n. All mice are located between the cat and the hole: the i-th mouse is located at the point x_i ($0 < x_i < n$). At each point, many mice can be located.

In one second, the following happens. First, **exactly one** mouse moves to the right by 1. If the mouse reaches the hole, it hides (i.e. the mouse will not any more move to any point and will not be eaten by the cat). Then (**after that** the mouse has finished its move) the cat moves to the right by 1. If at the new cat's position, some mice are located, the cat eats them (they will not be able to move after that). The actions are performed until any mouse hasn't been hidden or isn't eaten.

In other words, the first move is made by a mouse. If the mouse has reached the hole, it's saved. Then the cat makes a move. The cat eats the mice located at the pointed the cat has reached (if the cat has reached the hole, it eats nobody).

Each second, you can select a mouse that will make a move. What is the maximum number of mice that can reach the hole without being eaten?

Input

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

Each test case consists of two lines. The first line contains two integers n and k ($2 \le n \le 10^9$, $1 \le k \le 4 \cdot 10^5$). The second line contains k integers $x_1, x_2, \ldots x_k$ ($1 \le x_i < n$) — the initial coordinates of the mice.

It is guaranteed that the sum of all k given in the input doesn't exceed $4 \cdot 10^5$.

Output

For each test case output on a separate line an integer m ($m \ge 0$) — the maximum number of mice that can reach the hole without being eaten.

Example

input

```
3
10 6
8 7 5 4 9 4
2 8
1 1 1 1 1 1 1 1
12 11
1 2 3 4 5 6 7 8 9 10 11
```

output

```
3
1
4
```

D1. All are Same

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input

output: standard output

This problem is a simplified version of D2, but it has significant differences, so read the whole statement.

Polycarp has an array of n (n is even) integers $a_1, a_2, \ldots, a_n$. Polycarp conceived of a positive integer k. After that, Polycarp began performing the following operations on the array: take an index i ($1 \le i \le n$) and reduce the number a_i by k.

After Polycarp performed some (possibly zero) number of such operations, it turned out that **all** numbers in the array became the same. Find the maximum k at which such a situation is possible, or print -1 if such a number can be arbitrarily large.

Input
The first line contains one integer t ($1 \le t \le 10$) — the number of test cases. Then t test cases follow.

Each test case consists of two lines. The first line contains an even integer n ($4 \le n \le 40$) (n is even). The second line contains n integers $a_1, a_2, \ldots a_n$ ($-10^6 \le a_i \le 10^6$).

It is guaranteed that the sum of all n specified in the given test cases does not exceed 100.

Output
For each test case output on a separate line an integer k ($k \ge 1$) — the maximum possible number that Polycarp used in operations on the array, or -1, if such a number can be arbitrarily large.

Example

input
3 6 1 5 3 1 1 5 8 -1 0 1 -1 0 1 -1 0 4 100 -1000 -1000 -1000
output
2 1 1100

D2. Half of Same

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem is a complicated version of D1, but it has significant differences, so read the whole statement.

Polycarp has an array of n (n is even) integers $a_1, a_2, \ldots, a_n$. Polycarp conceived of a positive integer k. After that, Polycarp began performing the following operations on the array: take an index i ($1 \le i \le n$) and reduce the number a_i by k.

After Polycarp performed some (possibly zero) number of such operations, it turned out that **at least half** of the numbers in the array became the same. Find the maximum k at which such a situation is possible, or print -1 if such a number can be arbitrarily large.

Input
The first line contains one integer t ($1 \le t \le 10$) — the number of test cases. Then t test cases follow.

Each test case consists of two lines. The first line contains an even integer n ($4 \le n \le 40$) (n is even). The second line contains n integers $a_1, a_2, \ldots a_n$ ($-10^6 \le a_i \le 10^6$).

It is guaranteed that the sum of all n specified in the given test cases does not exceed 100.

Output
For each test case output on a separate line an integer k ($k \ge 1$) — the maximum possible number that Polycarp used in operations on the array, or -1, if such a number can be arbitrarily large.

Example

input
4 6 48 13 22 -15 16 35 8 -1 0 1 -1 0 1 -1 0 4 100 -1000 -1000 -1000 4 1 1 1 1
output
13 2

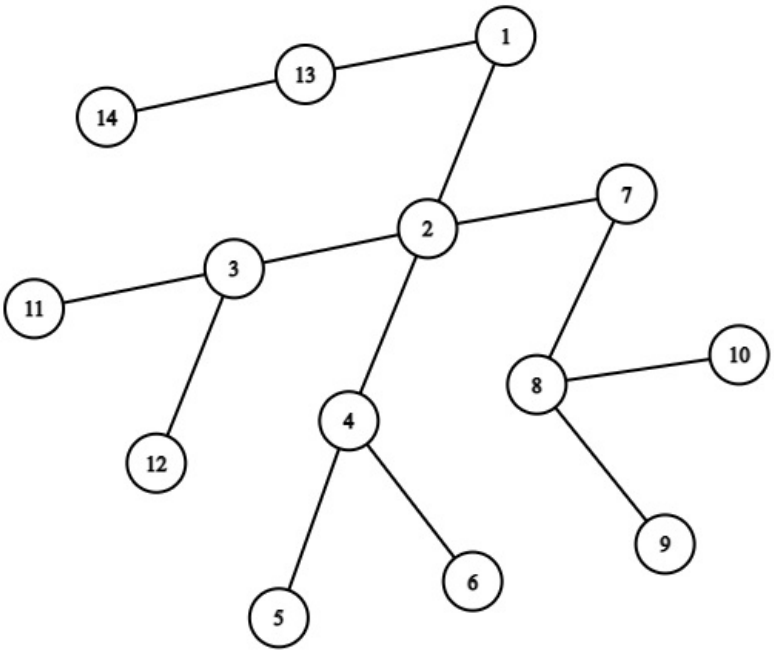
```
-1
-1
```

E. Gardener and Tree

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

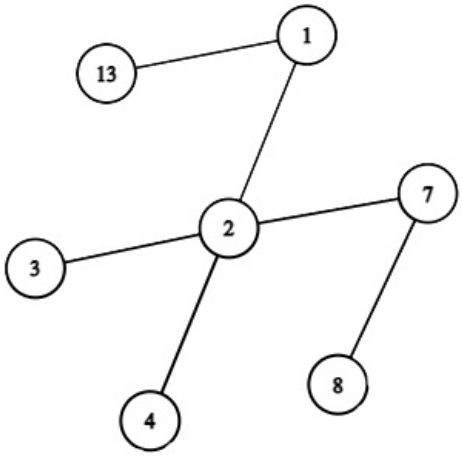
A tree is an undirected connected graph in which there are no cycles. This problem is about non-rooted trees. A leaf of a tree is a vertex that is connected to **at most one** vertex.

The gardener Vitaly grew a tree from n vertices. He decided to trim the tree. To do this, he performs a number of operations. In one operation, he removes **all** leaves of the tree.



Example of a tree.

For example, consider the tree shown in the figure above. The figure below shows the result of applying exactly one operation to the tree.



The result of applying the operation "remove all leaves" to the tree.

Note the special cases of the operation:

- applying an operation to an empty tree (of 0 vertices) does not change it;
- applying an operation to a tree of one vertex removes this vertex (this vertex is treated as a leaf);
- applying an operation to a tree of two vertices removes both vertices (both vertices are treated as leaves).

Vitaly applied k operations sequentially to the tree. How many vertices remain?

Input

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

Each test case is preceded by an empty line.

Each test case consists of several lines. The first line of the test case contains two integers n and k ($1 \le n \le 4 \cdot 10^5$, $1 \le k \le 2 \cdot 10^5$) — the number of vertices in the tree and the number of operations, respectively. Then $n - 1$ lines follow, each of them contains two integers u and v ($1 \le u, v \le n$, $u \ne v$) which describe a pair of vertices connected by an edge. It is guaranteed that the given graph is a tree and has no loops or multiple edges.

It is guaranteed that the sum of n from all test cases does not exceed $4 \cdot 10^5$.

Output

For each test case output on a separate line a single integer — the number of vertices that remain in the tree after applying k operations.

Example

input
6 14 1 1 2 2 3 2 4 4 5 4 6 2 7 7 8 8 9 8 10 3 11 3 12 1 13 13 14 2 200000 1 2 3 2 1 2 2 3 5 1 5 1 3 2 2 1 5 4 6 2 5 1 2 5 5 6 4 2 3 4 7 1 4 3 5 1 1 3 6 1 1 7 2 1
output
7 0 0 3 1 2

Note

The first test case is considered in the statement.

The second test case contains a tree of two vertices. 200000 operations are applied to it. The first one removes all two vertices, the other operations do not change the tree.

In the third test case, a tree of three vertices is given. As a result of the first operation, only 1 vertex remains in it (with the index 2), the second operation makes the tree empty.

F. Red-Black Number

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

It is given a non-negative integer x, the decimal representation of which contains n digits. You need to color **each** its digit in red or black, so that the number formed by the red digits is divisible by A, and the number formed by the black digits is divisible by B.

At least one digit must be colored in each of two colors. Consider, the count of digits colored in red is r and the count of digits colored in black is b. Among all possible colorings of the given number x, you need to output any such that the value of $|r - b|$ is **the minimum possible**.

Note that the number x and the numbers formed by digits of each color, **may contain leading zeros**.

Example of painting a number for $A = 3$ and $B = 13$

The figure above shows an example of painting the number $x = 02165$ of $n = 5$ digits for $A = 3$ and $B = 13$. The red digits form the number 015, which is divisible by 3, and the black ones — 26, which is divisible by 13. Note that the absolute value of the difference between the counts of red and black digits is 1, it is impossible to achieve a smaller value.

Input

The first line contains one integer t ($1 \le t \le 10$) — the number of test cases. Then t test cases follow.

Each test case consists of two lines. The first line contains three integers n, A, B ($2 \le n \le 40$, $1 \le A, B \le 40$). The second line contains a non-negative integer x containing exactly n digits and probably containing leading zeroes.

Output

For each test case, output in a separate line:

- -1 if the desired coloring does not exist;
- a string s of n characters, each of them is a letter 'R' or 'B'. If the i-th digit of the number x is colored in red, then the i-th character of the string s must be the letter 'R', otherwise the letter 'B'.

The number formed by digits colored red should divisible by A. The number formed by digits colored black should divisible by B. The value $|r - b|$ should be minimal, where r is the count of red digits, b is the count of black digits. If there are many possible answers, print any of them.

Example

input
4 5 3 13 02165 4 2 1 1357 8 1 1 12345678 2 7 9 90
output
RBRBR -1 BBRRRRBB BR

Note

The first test case is considered in the statement.

In the second test case, there are no even digits, so it is impossible to form a number from its digits that is divisible by 2.

In the third test case, each coloring containing at least one red and one black digit is possible, so you can color 4 digits in red and 4 in black ($|4 - 4| = 0$, it is impossible to improve the result).

In the fourth test case, there is a single desired coloring.

G. Changing Brackets

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A sequence of round and square brackets is given. You can change the sequence by performing the following operations:

1. change the direction of a bracket from opening to closing and vice versa without changing the form of the bracket: i.e. you can change '(' to ')' and ')' to '('; you can change '[' to ']' and ']' to '['. The operation costs 0 burles.
2. change any **square** bracket to **round** bracket having the same direction: i.e. you can change '[' to '(' but **not** from '(' to '['; similarly, you can change ']' to ')' but **not** from ')' to ']'. The operation costs 1 burle.

The operations can be performed in any order any number of times.

You are given a string s of the length n and q queries of the type "l r" where $1 \le l < r \le n$. For every substring $s[l \dots r]$, find the

minimum cost to pay to make it a correct bracket sequence. It is guaranteed that the substring $s[l \dots r]$ has an even length.

The queries must be processed independently, i.e. the changes made in the string for the answer to a question i don't affect the queries j ($j > i$). In other words, for every query, the substring $s[l \dots r]$ is given from the initially given string s.

A correct bracket sequence is a sequence that can be built according the following rules:

- an empty sequence is a correct bracket sequence;
- if "s" is a correct bracket sequence, the sequences "(s)" and "[s]" are correct bracket sequences.
- if "s" and "t" are correct bracket sequences, the sequence "st" (the concatenation of the sequences) is a correct bracket sequence.

E.g. the sequences "", "(( ) [ ])", "[( ) ( )] ( )" and "(( )) ( )" are correct bracket sequences whereas "(", "[( ])" and ")))" are not.

Input
The first line contains one integer t ($1 \le t \le 100$) — the number of test cases. Then t test cases follow.

For each test case, the first line contains a non-empty string s containing only round ('(', ')') and square ('[', ']') brackets. The length of the string doesn't exceed 10^6. The string contains at least 2 characters.

The second line contains one integer q ($1 \le q \le 2 \cdot 10^5$) — the number of queries.

Then q lines follow, each of them contains two integers l and r ($1 \le l < r \le n$ where n is the length of s). It is guaranteed that the substring $s[l \dots r]$ has even length.

It is guaranteed that the sum of the lengths of all strings given in all test cases doesn't exceed 10^6. The sum of all q given in all test cases doesn't exceed $2 \cdot 10^5$.

Output
For each test case output in a separate line for each query one integer x ($x \ge 0$) — the minimum cost to pay to make the given substring a correct bracket sequence.

Example

input
3 ([))[]()][[]] 3 1 12 4 9 3 6)))))) 2 2 3 1 4 [] 1 1 2
output
0 2 1 0 0 0

Note
Consider the first test case. The first query describes the whole given string, the string can be turned into the following correct bracket sequence: "([( )]) ( ) [[ ]]". The forms of the brackets aren't changed so the cost of changing is 0.

The second query describes the substring ") [ ) ( )]". It may be turned into "(( ) ( ))", the cost is equal to 2.

The third query describes the substring ")) [ )". It may be turned into "( ) ( )", the cost is equal to 1.

The substrings of the second test case contain only round brackets. It's possible to prove that any sequence of round brackets having an even length may be turned into a correct bracket sequence for the cost of 0 burles.

In the third test case, the single query describes the string "[ ]" that is already a correct bracket sequence.