Problem A. Your Shine Your Be!

Input file: `standard input`
Output file: `standard output`

You are given a sequence of non-negative integers $a_1, a_2, \ldots, a_n$ of length n.

There are q queries. Each query provides an interval $[l, r]$. For each query, you need to consider the multiset V formed by the elements in the interval $[l, r]$, and select a subset $M \subseteq V$ of maximum size such that all elements in M have pairwise distinct values.

After that, you may choose any subset $T \subseteq M$ (including the empty set). Let $|M|$ denote the size of set M, and let X_T denote the bitwise XOR of all elements in T (the XOR of the empty set is defined as 0). Your task is to compute the minimum possible value of $|M| \oplus X_T$.

This problem is online. The query inputs l', r' are encrypted. Let the answer to the previous query be $lastans$ (initially $lastans = 0$). The actual query endpoints l, r are obtained as follows:

$$l = (l' \oplus lastans) \bmod n + 1$$

$$r = (r' \oplus lastans) \bmod n + 1$$

After decryption, if $l > r$, you should swap l and r.

Input

The first line contains two integers n, q ($1 \le n, q \le 5 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le n$).

The following q lines each contain two integers l', r' ($0 \le l', r' \le 2 \cdot 10^9$), representing the encrypted query interval endpoints.

Output

For each query, output one integer on a separate line representing the answer.

Example

standard input	standard output
6 3 4 4 6 6 0 0 6 6 2 4 6 3	1 2 1

Note

Initially, $lastans = 0$.

- For the first query, after decryption we get $l = 1, r = 1$. The set of elements in the interval is $S = \{4\}$, and $|S| = 1$. Choose the empty set $\varnothing$ (whose XOR sum is 0), then the answer is $1 \oplus 0 = 1$. Update $lastans = 1$.
- For the second query, after decryption we get $l = 4, r = 6$. The set of elements in the interval is $S = \{0, 6\}$, and $|S| = 2$. Choose the empty set $\varnothing$ (whose XOR sum is 0), then the answer is $2 \oplus 0 = 2$. Update $lastans = 2$.
- For the third query, after decryption we get $l = 5, r = 2$, so we swap the endpoints and obtain $[2, 5]$. The set of elements in the interval is $S = \{0, 4, 6\}$, and $|S| = 3$. Choose the subset $\{4, 6\}$ (whose XOR sum is 2), then the answer is $3 \oplus 2 = 1$. Update $lastans = 1$.

The input and output size of this problem is large, so it is recommended to use fast I/O methods. For C++ contestants, it is recommended to use `scanf/printf`, or disable synchronization when using `cin/cout`:

```
std::ios::sync_with_stdio(false);
std::cin.tie(nullptr);
```

Problem B. Bus Game

Input file: `standard input`
Output file: `standard output`

There are n consecutive bus seats arranged in a line, numbered from 1 to n. Due to long-term disrepair, m of these seats are broken.

Alice and Bob take turns seating passengers, with Alice moving first. In each move, the current player must choose an unbroken and currently empty seat i for a passenger, subject to the following condition:

All seats j satisfying $|j - i| \leq k$ and $j \neq i$ must be empty (that is, all other seats within distance at most k from i must be unoccupied. Note that broken seats do not affect this distance check; the distance is still computed across broken seats).

After the passenger sits down, seat i becomes occupied. If the current player can no longer find any legal empty seat for seating a passenger, the game ends immediately. Alice wants the total number of seated passengers to be as large as possible, while Bob wants it to be as small as possible. Both players play optimally.

Please output the final possible seat configuration of the bus as a string.

Input

The first line contains three integers n, k, and m ($1 \leq n \leq 20$, $1 \leq k \leq 20$, $0 \leq m \leq n$), representing the total number of seats, the distance constraint, and the number of broken seats, respectively.

The second line contains m distinct integers, indicating the indices of the broken seats.

Output

For each test case, output a string of length n on a single line, representing the final seat configuration of the bus:

- Character `'a'` indicates that the seat is occupied by a passenger seated by Alice.
- Character `'b'` indicates that the seat is occupied by a passenger seated by Bob.
- Character `'x'` indicates that the seat is broken (and cannot be occupied).
- Character `'.'` indicates that the seat is intact but remains empty in the end.

If there are multiple possible results, output any one of them.

Examples

standard input	standard output
1 1 0	a
5 2 1 3	a.xb.

Note

In the first sample, Alice seats a passenger at seat 1. After that, there is no legal empty seat left, so Bob cannot make a move and the game ends immediately. The final state is `a`.

In the second sample, seat 3 is broken, so the initial state is `..x..`

- Alice may choose seat 1, making the state `a.x..` Since the distance constraint is $k = 2$, seat 2 then becomes unavailable.

- Bob may choose seat 4, making the state `a.xb.`. Similarly, seat 5 then becomes unavailable.
- Neither player has any legal seat to choose, so the game ends.

In this playthrough, both players play optimally. Note that `a.x.b` is also a valid and correct final state.

Problem C. Crossing

Input file: standard input
Output file: standard output

The Ao–Tai Line is a high-altitude hiking route connecting Aoshan and Taibai Mountain in the Qinling Mountains. It is famous for its unpredictable weather, steep terrain, and thin oxygen, and is known as a dangerous route that "walks along the backbone of China."Along this route, each type of terrain corresponds to a different level of expedition difficulty and resource supply.

As an experienced explorer, Fanfan needs to plan a safe path from the starting point to the destination. Whenever Fanfan reaches a location, a certain amount of stamina is consumed, and some supplies are either consumed or replenished. We use an $n \times m$ mountain map to model this route, where each location is marked with a value.

You need to start from the upper-left corner $(1, 1)$ and reach the lower-right corner (n, m). You may move in the four directions: up, down, left, and right, and you may visit the same location multiple times.

Suppose the values of the locations along the explorer's path are $a_1, a_2, \ldots, a_k$, where a_1 is the value at the starting point $(1, 1)$ and a_k is the value at the ending point (n, m). Define:

The stamina consumed as $OR = a_1 \mid a_2 \mid \ldots \mid a_k$ (bitwise OR),

The supplies remaining as $AND = a_1 \& a_2 \& \ldots \& a_k$ (bitwise AND).

Your task is to minimize the stamina consumed, and among all such paths, maximize the supplies remaining. Output the minimum stamina consumption and the maximum supplies remaining satisfying the condition.

Input

The first line contains an integer T $(1 \le T \le 100)$, denoting the number of test cases.

For each test case:

The first line contains two integers n and m $(1 \le n, m \le 10^5$ and $n \times m \le 2 \cdot 10^5)$, denoting the number of rows and columns of the matrix.

The next n lines each contain m non-negative integers $a_{ij} < 2^{30}$, representing the elements in the matrix.

It is guaranteed that for each test file, the sum of $n \times m$ over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output one line containing two integers, representing the minimum stamina consumption and the maximum supplies remaining that satisfy the conditions.

Examples

standard input	standard output
1 2 2 11 2 3 7	15 3
1 2 1 14 100	110 4

Note

For the first sample:

The route $(1,1) \to (1,2) \to (2,2)$ has stamina consumption 15 and supplies remaining 2;

The route $(1,1) \to (2,1) \to (2,2)$ has stamina consumption 15 and supplies remaining 3;

There is no other route with smaller stamina consumption, so we choose the one with more supplies remaining. Therefore, the answer is 15 3.

Problem D. Taffy's LCM

Input file: `standard input`
Output file: `standard output`

Taffy gives you an integer c.

You need to help Taffy find two positive integers a and b such that $a + b = c$, and their least common multiple $\text{lcm}(a, b)$ is maximized.

The least common multiple of two positive integers a and b is the smallest positive integer that is divisible by both a and b.

Input

The input contains multiple test cases.

The first line contains an integer t ($1 \le t \le 10^4$).

Each test case consists of a single line containing one integer c ($2 \le c \le 10^9$).

Output

For each test case, output one integer — the maximum possible value of $\text{lcm}(a, b)$.

Example

standard input	standard output
4 4 6 9 114514	3 5 20 3278364045

Problem E. Simple Math

Input file: `standard input`
Output file: `standard output`

As everyone knows, Cirno's favorite number is 9!

Today, Cirno gets a positive integer x. She suddenly wonders: can she find another positive integer k such that their product $x \cdot k$ becomes a number whose every digit is 9 (for example, 9, 99, 999999, and so on)?

Input

The input contains multiple test cases.

The first line contains an integer t ($1 \le t \le 10^4$), denoting the number of test cases.

For each test case, the first line contains a positive integer x ($1 \le x \le 10^{12}$).

It is guaranteed that the sum of all x over a single test file does not exceed 10^{12}.

Output

For each test case:

If there exists an integer k satisfying the condition, output `YES` on a single line.

Otherwise, output `NO` on a single line.

You may print the answer in any letter case. For example, the strings `yEs`, `yes`, and `Yes` will all be accepted as affirmative answers.

Example

standard input	standard output
4 3 10 7 12	YES NO YES NO

Note

- **In the first test case**, $x = 3$. If we choose $k = 3$, then their product is $3 \times 3 = 9$, which satisfies the condition. Therefore, the answer is `YES`.
- **In the second test case**, $x = 10$. It can be shown that there does not exist any positive integer k such that $10 \cdot k$ consists entirely of the digit 9. Therefore, the answer is `NO`.
- **In the third test case**, $x = 7$. If we choose $k = 142857$, then their product is $7 \times 142857 = 999999$, which satisfies the condition. Therefore, the answer is `YES`.
- **In the fourth test case**, $x = 12$. It can be shown that there does not exist any positive integer k such that every digit of the product is 9. Therefore, the answer is `NO`.

Problem F. Fanfan's Bracket Sequence

Input file: standard input
Output file: standard output

Fanfan has a valid bracket sequence s containing n bracket pairs, and all bracket pairs are numbered from 1 to n. The numbering is assigned according to the order in which the left brackets appear: the first left bracket that appears and its matching right bracket form bracket pair 1, the second left bracket that appears and its matching right bracket form bracket pair 2, and so on.

Based on this valid bracket sequence, Fanfan constructs a binary tree according to the following rules:

For the bracket pair numbered p (whose corresponding bracket positions in the sequence are s_i and s_j, where s_i is a left bracket and s_j is the matching right bracket), we examine s_{i+1} and s_{j+1}:

If s_{i+1} is a left bracket, then let the corresponding bracket pair be numbered l, and make l the left child of p;

If s_{j+1} is a left bracket, then let the corresponding bracket pair be numbered r, and make r the right child of p.

Now, Fanfan has lost the original bracket sequence s, but the constructed binary tree structure is still available. Your task is to reconstruct the original valid bracket sequence s from the given binary tree.

Input

The first line contains an integer t ($1 \le t \le 10^4$), denoting the number of test cases.

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

The next n lines each contain two integers l and r, representing the left child and right child of each bracket pair numbered from 1 to n, respectively. If a bracket pair has no left child or no right child, then the corresponding l or r is 0.

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

Output

For each test case, output a line containing a string consisting of (and), representing the reconstructed valid bracket sequence s.

Example

standard input	standard output
1 3 2 3 0 0 0 0	(())()

Note

The binary tree structure corresponding to the sample input is as follows:

The bracket pair numbered 1 has left child 2 and right child 3;

The bracket pairs numbered 2 and 3 have no children.

The corresponding valid bracket sequence is (())(), where:

Bracket pair 1 consists of the 1st character (a left bracket) and the 4th character (a right bracket);

Bracket pair 2 consists of the 2nd character (a left bracket) and the 3rd character (a right bracket);

Bracket pair 3 consists of the 5th character (a left bracket) and the 6th character (a right bracket).

Problem G. Taffy vs Goblins

Input file: `standard input`
Output file: `standard output`

Mage Taffy is fighting against N goblins. She will cast N magic missiles, where the i-th missile is fixed to attack the i-th goblin.

It is known that the i-th missile has penetration power A_i and potential damage C_i, while the i-th goblin has defense B_i. The missile deals C_i damage if and only if its penetration power is at least the goblin's defense, that is, $A_i \geq B_i$; otherwise, it deals 0 damage.

Taffy has one chance to use a secret spell: before damage is resolved, she may choose **at most** one missile and forcibly change its penetration power to X (or she may choose not to modify any missile).

Now there are Q **independent** queries. Each query gives a value X. For the current given X, please determine the maximum total damage Taffy can deal to the goblins under the optimal modification strategy.

Input

The input contains multiple test cases.

The first line contains an integer t ($1 \leq t \leq 10^4$), denoting the number of test cases.

For each test case:

The first line contains two integers N, Q ($1 \leq N, Q \leq 2 \cdot 10^5$), denoting the number of goblins (and missiles) and the number of queries, respectively.

The second line contains N integers $A_1, A_2, \ldots, A_N$.

The third line contains N integers $B_1, B_2, \ldots, B_N$.

The fourth line contains N integers $C_1, C_2, \ldots, C_N$.

The next Q lines each contain one integer X, representing the given modification value for a query.

Constraints:

- $1 \leq A_i, B_i, C_i, X \leq 10^9$
- It is guaranteed that for each test file, the sum of all N and the sum of all Q over all test cases do not exceed $2 \cdot 10^5$, respectively.

Output

For each test case, output Q lines. Each line should contain one integer, representing the maximum total damage Taffy can deal to the goblins for the corresponding modification value X.

Example

standard input	standard output
1 5 4 1 2 4 5 6 3 1 5 8 7 1 5 5 2 1 3 1 7 8	6 5 10 10

Note

Initially, without using the secret spell, only the 2nd missile hits, so the base damage is 5.

For the first query $X = 3$: modify the 1st missile. Since $3 \geq B_1$, it deals an additional 1 point of damage. The maximum total damage is $5 + 1 = 6$.

For the second query $X = 1$: modifying any missile that originally misses still cannot make it satisfy the hit condition, so not modifying any missile is optimal. The maximum total damage remains the base value 5.

For the third query $X = 7$: modifying the 3rd missile is optimal. Since $7 \geq B_3$, it deals an additional 5 points of damage. The maximum total damage is $5 + 5 = 10$.

For the fourth query $X = 8$: it is still optimal to modify the 3rd missile (for a total damage of 10); if the 4th missile were modified instead, the total damage would only be $5 + 2 = 7$. Therefore, the maximum total damage is still 10.

Problem H. Little t's Recurrence

Input file: `standard input`
Output file: `standard output`

Please note the special time limit and memory limit for this problem.

Little t has recently become very interested in fascinating number sequences and binary representations.

He defines a generalized recurrence sequence $f(i)$ with parameter a, formally given by:

$$f(i) = \begin{cases} 0 & \text{if } i = 0, \\ 1 & \text{if } i = 1, \\ a \cdot f(i-1) + f(i-2) & \text{if } i \geq 2. \end{cases}$$

At the same time, Little t defines the function $g(x)$ as the number of `1`s in the binary representation of a positive integer x (that is, the popcount). For example, $g(5) = g(101_{(2)}) = 2$.

Now, Little t has an extremely large positive integer n. Since it is too large, he can only write it as a binary string s without leading zeros. He is also given the parameter a of the recurrence sequence.

He really wants to know the value of the following summation:

$$\sum_{i=1}^{n} f(g(i)) \pmod{10^9 + 7}$$

However, since the number n is far too large, Little t's computer was unable to compute the answer. Can you write a program to help him?

Input

The first line contains a string s consisting only of `0` and `1` ($1 \leq |s| \leq 10^7$), representing the binary form of the number n. It is guaranteed that the first character is `1`.

The second line contains a positive integer a ($1 \leq a \leq 10^9$).

Output

Output one integer on a single line, representing the value of the summation required by Little t modulo $10^9 + 7$.

Example

standard input	standard output
101 2	7

Note

In the sample, the string `101` corresponds to the decimal number $n = 5$. The parameter is $a = 2$.

The first few terms of the sequence f are computed as follows:

- $f(0) = 0$
- $f(1) = 1$
- $f(2) = 2 \times 1 + 0 = 2$

We need to compute the sum of $f(g(i))$ for i from 1 to 5:

- $i = 1$ ($\texttt{1}_{(2)}$): $g(1) = 1 \implies f(1) = 1$
- $i = 2$ ($\texttt{10}_{(2)}$): $g(2) = 1 \implies f(1) = 1$
- $i = 3$ ($\texttt{11}_{(2)}$): $g(3) = 2 \implies f(2) = 2$
- $i = 4$ ($\texttt{100}_{(2)}$): $g(4) = 1 \implies f(1) = 1$
- $i = 5$ ($\texttt{101}_{(2)}$): $g(5) = 2 \implies f(2) = 2$

Therefore, the final answer is $1 + 1 + 2 + 1 + 2 = 7$.

The input and output size of this problem is large, so it is recommended to use fast I/O methods. For C++ contestants, it is recommended to use `scanf/printf`, or disable synchronization when using `cin/cout`:

```
std::ios::sync_with_stdio(false);
std::cin.tie(nullptr);
```

Problem I. Milk Tea

Input file: `standard input`
Output file: `standard output`

On Sunday afternoon, the HZAU ACM training room is filled with the sound of keyboards. To reward the students who are working hard on programming problems, Fan Ge decides to order some milk tea from Mixue Bingcheng.

There are n students in the training room, and Fan Ge needs to buy **exactly** n cups of milk tea.

After opening the food delivery app, Fan Ge finds that the shop offers two ways to buy:

- Single order: buying 1 cup of milk tea costs a yuan.
- Couple set: buying 2 cups of milk tea costs only b yuan.

Now, please help Fan Ge calculate the minimum amount of money needed to buy **exactly** n cups of milk tea.

Input

The input contains multiple test cases.

The first line contains an integer t ($1 \le t \le 10^4$), denoting the number of test cases.

For each test case, one line contains three integers n, a, b ($1 \le n, a, b \le 10^9$), representing the required number of cups of milk tea, the price of a single cup, and the price of a couple set, respectively.

Output

For each test case, output one integer representing the minimum cost to buy exactly n cups of milk tea.

Example

standard input	standard output
4 5 2 5 4 3 4 5 3 4 1 10 5	10 8 11 10

Note

In the first test case, the optimal strategy is to buy all cups individually. The minimum cost to buy 5 cups is $5 \times 2 = 10$ yuan. **In the second test case**, the optimal strategy is to buy 2 couple sets. The minimum cost is $2 \times 4 = 8$ yuan. **In the third test case**, the optimal strategy is to buy one couple set and one single cup. The minimum cost is $2 \times 4 + 1 \times 3 = 11$ yuan. **In the fourth test case**, the minimum cost is 10 yuan.

Problem J. Tree

Input file: `standard input`
Output file: `standard output`

You are given a rooted tree with root 1 and N nodes. Each node i has an initial weight M_i. At all times, the system strictly maintains the following property: for any non-root node x, its weight never exceeds that of its parent, that is, $M_{\text{parent}(x)} \geq M_x$.

You need to process Q operations online. The operations are of the following two types:

- `1 u w`: **Query**. Starting from node u, move upward along the tree edges toward the root, and find the first node whose weight is at least w (including u itself). Output the index of that node. If no such node exists all the way up to the root, output `-1`.
- `2 x v`: **Update**. Attempt to add v to the weight of node x (v may be negative), that is, let $M_x \leftarrow M_x + v$. To avoid violating the tree property, a legality check must be performed before the update:
  - If $x \neq 1$ and $M_x + v > M_{\text{parent}(x)}$, then the update fails.
  - If x is not a leaf node and $M_x + v < \max_{y \in \text{children}(x)} M_y$, then the update fails.

  If both checks pass, then the update is actually performed and `SUCCESS` is output; otherwise, the update is rejected, the state of the tree remains unchanged, and `FAILED` is output.

Input

The input contains multiple test cases.

The first line contains an integer t ($1 \leq t \leq 10^4$), denoting the number of test cases.

For each test case, the first line contains two integers N, Q.

The second line contains N integers $M_1, M_2, \ldots, M_N$, and it is guaranteed that the initial weights satisfy the property described above.

The next $N - 1$ lines each contain two integers u, v, representing an edge in the tree.

The following Q lines each contain three integers opt, a, b, representing an operation: when $opt = 1$, it denotes a `1 u w` operation; when $opt = 2$, it denotes a `2 x v` operation.

Constraints:

- $1 \leq N, Q \leq 10^5$
- $0 \leq M_i, w \leq 10^9$
- $-10^9 \leq v \leq 10^9$
- It is guaranteed that $M_i \geq 0$ at all times

It is guaranteed that for each test file, the sum of all N and the sum of all Q over all test cases do not exceed $2 \cdot 10^5$, respectively.

Output

For each test case, output Q lines, each corresponding to the result of one operation:

- For a `1 u w` operation, output the index of the node satisfying the condition; if no such node exists, output `-1`.
- For a `2 x v` operation, output `SUCCESS` if the update succeeds, and `FAILED` if the update fails.

Example

standard input	standard output
1 5 6 100 80 50 30 20 1 2 1 3 2 4 2 5 1 4 40 1 5 150 2 2 30 2 2 -60 2 4 10 1 4 40	2 -1 FAILED FAILED SUCCESS 4

Note

In the sample, the tree structure is as follows: node 1 has children 2 and 3; node 2 has children 4 and 5. The initial weights are $M = [100, 80, 50, 30, 20]$.

- **Operation 1** `1 4 40`: starting from node 4, we move upward to find a node whose weight is at least 40. The current weight of node 4 is $30 < 40$, while its parent node 2 has weight $80 \geq 40$, which satisfies the condition, so the output is `2`.
- **Operation 2** `1 5 150`: starting from node 5, we move upward to find a node whose weight is at least 150. None of the nodes on the path, including the root node 1 (whose weight is 100), satisfies the condition, so the output is `-1`.
- **Operation 3** `2 2 30`: we attempt to increase the weight of node 2 by 30, so the new weight would be 110. Since $110 > M_1$ (100), this violates the property that a parent's weight must be at least that of its child, so the update fails and the output is `FAILED`.
- **Operation 4** `2 2 -60`: we attempt to decrease the weight of node 2 by 60, so the new weight would be 20. Since $20 < \max(M_4, M_5)$ (30), this also violates the monotonicity property of the tree weights, so the update fails and the output is `FAILED`.
- **Operation 5** `2 4 10`: we attempt to increase the weight of node 4 by 10, so the new weight becomes 40. This satisfies $40 \leq M_2$ (80), and node 4 is a leaf node, so the update succeeds. We update $M_4 = 40$ and output `SUCCESS`.
- **Operation 6** `1 4 40`: again starting from node 4, we move upward to find a node whose weight is at least 40. Since the previous update succeeded, we now have $M_4 = 40 \geq 40$, so node 4 itself already satisfies the condition, and the output is `4`.

Problem K. MEX

Input file: standard input
Output file: standard output

You are given an undirected tree with n nodes, numbered from 1 to n. Each node i on the tree has a weight p_i. It is guaranteed that the weights $p_1, p_2, \ldots, p_n$ form a permutation.

For a connected component on the tree (that is, a connected subgraph consisting of some nodes of the tree and the edges between them), we define its MEX value as the smallest non-negative integer that does not appear in the set of node weights within this connected component.

Now there are q queries. Each query gives two integers A and B. For each query, you need to determine how many connected components in the tree satisfy both of the following conditions:

1. The connected component contains node A.
2. The MEX value of the connected component is exactly B.

Since the number of connected components satisfying the conditions may be very large, output the answer to each query modulo $10^9 + 7$.

- **Permutation**: If a sequence of length n contains every integer from 0 to $n - 1$ exactly once, then it is called a permutation of 0 to $n - 1$. For example, $[2, 0, 3, 1]$ is a permutation, while $[0, 1, 1, 3]$ is not.

Input

The input contains multiple test cases.

The first line contains an integer t ($1 \le t \le 10^4$), denoting the number of test cases.

For each test case, the first line contains two integers n, q ($1 \le n, q \le 2 \cdot 10^5$).

The second line contains n integers $p_1, p_2, \ldots, p_n$ ($0 \le p_i < n$). It is guaranteed that these numbers form a permutation of 0 to $n - 1$.

The next $n - 1$ lines each contain two integers u, v ($1 \le u, v \le n$), representing an edge in the tree.

The following q lines each contain two integers A, B ($1 \le A \le n, 0 \le B \le n$), representing a query.

It is guaranteed that for each test file, the sum of all n and the sum of all q over all test cases do not exceed $2 \cdot 10^5$, respectively.

Output

For each test case, output q lines. Each line should contain one integer, representing the answer to the corresponding query modulo $10^9 + 7$.

Example

standard input	standard output
1 5 5 2 0 1 3 4 1 2 2 3 3 4 3 5 2 0 4 1 4 2 1 3 2 4	0 0 2 2 1

Note

In the first sample:

- For the first query ($A = 2, B = 0$): node 2 has weight 0. Any connected component containing node 2 must have MEX value greater than 0, so there is no connected component satisfying the conditions.
- For the second query ($A = 4, B = 1$): any connected component satisfying the conditions must contain node 4 and node 2 (whose weight is 0), but must not contain node 3 (whose weight is 1). Since the unique path connecting node 4 and node 2 in the tree must pass through node 3, it is impossible to construct such a valid connected component.
- For the third query ($A = 4, B = 2$): the connected components satisfying the conditions are $\{2, 3, 4\}$ and $\{2, 3, 4, 5\}$. Both have MEX value 2 and both contain node 4, so there are 2 valid choices.
- For the fourth query ($A = 1, B = 3$): the connected components satisfying the conditions are $\{1, 2, 3\}$ and $\{1, 2, 3, 5\}$, so there are 2 valid choices.

Problem L. Tornado Destroys the Parking Lot

Input file: standard input
Output file: standard output

You are assigned to manage a severely congested grid parking lot. The parking lot can be viewed as an $N \times M$ matrix. Each cell in the matrix is either empty (denoted by .), or occupied by a car with a fixed driving direction, which can be one of the four types: U (up), D (down), L (left), or R (right).

You may send a "leave" signal to these cars in any order. After receiving the signal, a car will successfully leave the parking lot if and only if, starting from its current position and moving straight toward the boundary in the direction it is facing, there is no other car blocking its path. Once a car leaves, its current cell becomes empty (.).

Determine whether there exists an order of sending signals such that all cars in the parking lot can be evacuated. If it is possible, output the sequence of coordinates of the cars to which signals are sent in order; otherwise, if the cars block each other in a deadlock and cannot all be evacuated, output -1.

Input

The first line contains two integers N and M ($1 \le N, M \le 2000$), representing the number of rows and columns of the parking lot, respectively.

The next N lines each contain a string of length M, consisting only of the characters U, D, L, R, and ..

Output

If it is possible to evacuate all cars, output K lines, where K is the total number of cars initially in the parking lot. Each line should contain two integers r and c ($1 \le r \le N$, $1 \le c \le M$), indicating the row and column of the car to which the signal is sent, in order (coordinates are 1-indexed). If there are multiple valid evacuation orders, output any one of them.

If it is impossible to evacuate all cars regardless of the order, output a single line containing -1.

Examples

standard input	standard output
3 3 D.. R.. .UL	2 1 3 2 1 1 3 3
2 2 RD UL	-1

Note

In the first sample, we can evacuate the cars in the following order:

1. Send a signal to the car at $(2, 1)$ facing right (R). Since there is no other car blocking its straight path to the right boundary, it successfully leaves.

2. Send a signal to the car at $(3, 2)$ facing up (U). Its path upward is clear, so it successfully leaves.

3. Send a signal to the car at $(1, 1)$ facing down (D). The car at $(2, 1)$ that originally blocked it has already left, so it can now leave successfully.

4. Send a signal to the car at $(3, 3)$ facing left (L). The car at $(3, 2)$ that originally blocked it has already left, so it also leaves successfully.

At this point, all cars have been successfully evacuated. Note that there may be more than one valid output order.

In the second sample, the car at $(1, 1)$ is blocked by the car at $(1, 2)$; the car at $(1, 2)$ is blocked by the car at $(2, 2)$; the car at $(2, 2)$ is blocked by the car at $(2, 1)$; and the car at $(2, 1)$ is in turn blocked by the car at $(1, 1)$. Initially, no car can move, so a deadlock occurs and complete evacuation is impossible. Therefore, the answer is -1.

Problem M. Little t's GCD

Input file: `standard input`
Output file: `standard output`

Little t gives you a sequence of n positive integers $a_1, a_2, \ldots, a_n$, as well as a constant P. You need to partition this sequence into m ($1 \le m \le n$) consecutive and non-empty subsegments. Each element must belong to exactly one subsegment. Suppose one of the subsegments you obtain is $[L, R]$ ($1 \le L \le R \le n$). For any given subsegment $[L, R]$, we define $W(L, R)$ as the sum of all positive odd numbers corresponding to the indices in this interval. Formally,

$$W(L,R) = \sum_{i=L}^{R} (2i-1)$$

At the same time, define $G(L, R)$ as the greatest common divisor of all elements in this interval. Formally,

$$G(L,R) = \gcd(a_L, a_{L+1}, \ldots, a_R)$$

Each time you create a subsegment, a cost of P is incurred. Therefore, for the interval $[L, R]$, its weight is calculated as follows:

$$\mathcal{E}(L,R) = W(L,R) \times G(L,R) - P$$

Little t wants you to find a partition scheme such that the total weight of these m subsegments is as large as possible.

Input

The input contains multiple test cases.

The first line contains an integer t ($1 \le t \le 10^4$), denoting the number of test cases.

For each test case, the first line contains two integers n, P ($1 \le n \le 10^5$, $0 \le P \le 10^{14}$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^5$).

It is guaranteed that the sum of all n over a single test file does not exceed 10^5.

Output

For each test case, output one integer representing the answer.

Example

standard input	standard output
1 3 25 8 12 6	6