Problem A
Storing Eggs

You have an egg carton that can be represented as a $3 \times N$ grid. The grid consists of 3 rows, numbered from 1 to 3, and N columns, numbered from 1 to N. The cell at row r and column c is denoted as (r, c). Each cell can be either usable or unusable; each usable cell can only hold at most 1 egg while unusable cells, as the name implies, cannot be used.

You want to put exactly K eggs into usable cells of your carton such that the distance between any two closest eggs is maximized. The distance between an egg in cell (r_1, c_1) and another egg in cell (r_2, c_2) can be calculated using Euclidean distance, i.e. $\sqrt{(r_1 - r_2)^2 + (c_1 - c_2)^2}$.

Determine the maximum possible distance between any two closest eggs, or determine if it is impossible to put K eggs into your carton.

Input

Input begins with two integers N K ($1 \leq N \leq 100$; $2 \leq K \leq 3N$) representing the number of columns of your egg carton and the number of eggs. Each of the next 3 lines contains a string S_r of length N that consists of either character '`.`' or '#'. The c^{th} character of string S_r represents the condition of cell (r, c) of the carton. Cell (r, c) is usable if $S_{r,c} =$ '`.`' and unusable if $S_{r,c} =$ '#'.

Output

If K eggs can be put into your carton, then output a real number in a single line representing the maximum possible distance between any two closest eggs. Your answer is considered correct if its absolute or relative error does not exceed 10^{-6}.

If K eggs cannot be put into your carton, then output `-1` in a single line.

Sample Input #1

```
5 2
#....
.....
....#
```

Sample Output #1

```
4.472136
```

Explanation for the sample input/output #1

The maximum distance between any two closest eggs can only be achieved by putting the eggs in cells $(3, 1)$ and $(1, 5)$, where the distance between the two (closest) eggs is $\sqrt{20}$.

Sample Input #2

```
5 6
##.##
#####
.....
```

Sample Output #2

```
1.000000
```

Explanation for the sample input/output #2

There is only one way to put 6 eggs into the carton; the distance between the two closest eggs is 1.

Sample Input #3

```
3 4
..#
...
...
```

Sample Output #3

```
1.414214
```

Explanation for the sample input/output #3

The maximum distance between any two closest eggs can be achieved by putting the eggs in cells $(1, 1)$, $(3, 1)$, $(3, 3)$, and $(2, 2)$. In this arrangement, the distance of the two closest eggs is $\sqrt{2}$, e.g., between eggs in cells $(1, 1)$ and $(2, 2)$.

Another way to put the eggs while getting the same answer is by putting the eggs in cells $(1, 2)$, $(2, 1)$, $(2, 3)$, and $(3, 2)$.

Sample Input #4

```
2 6
..
.#
..
```

Sample Output #4

```
-1
```

Explanation for the sample input/output #4

There are only 5 usable cells, thus, it is impossible to put 6 eggs into the carton.

Problem B
Magical Barrier

There are N power sources, numbered from 1 to N, scattered around the ICPC Kingdom. Power source i is uniquely located at coordinate (X_i, Y_i) in a 2D Cartesian plane such that there are no three power sources located in a straight line.

For each pair of distinct power sources i and j that satisfies $1 \leq i < j \leq N$, a magical barrier forms as a line segment that spans from (X_i, Y_i) to (X_j, Y_j).

You noticed a strange phenomenon. When two distinct magical barriers are intersecting, then both magical barriers are somewhat strengthened. To simplify things, you define the **strength** of a magical barrier b as the number of magical barriers other than b that intersects with b. Two distinct magical barriers are intersecting if and only if there exists exactly one point (x, y) that lies on both magical barriers while none of the N power sources are located at (x, y).

You want to find the strength of the strongest magical barrier in the ICPC Kingdom.

Input

Input begins with an integer N ($2 \leq N \leq 1000$) representing the number of power sources. Each of the next N lines contains 2 integers X_i Y_i ($-10^9 \leq X_i, Y_i \leq 10^9$) representing the location of power source i. It is guaranteed that the location of each power source is unique, and there are no three power sources located in a straight line.

Output

Output an integer in a single line representing the strength of the strongest magical barrier.

Sample Input #1

```
6
0 0
0 6
6 0
6 6
1 4
1 2
```

Sample Output #1

```
3
```

Explanation for the sample input/output #1

Let $\langle i, j \rangle$ be the magical barrier that spans from power source i to power source j.

One of the strongest magical barriers is $\langle 1, 4 \rangle$ with a strength of 3. The 3 magical barriers that intersect with $\langle 1, 4 \rangle$ are $\langle 2, 3 \rangle$, $\langle 3, 6 \rangle$, and $\langle 3, 5 \rangle$. Note that the magical barrier $\langle 2, 3 \rangle$ also has a strength of 3.

Sample Input #2

```
2
0 0
0 1
```

Sample Output #2

```
0
```

Explanation for the sample input/output #2

The only magical barrier is $\langle 1, 2 \rangle$ with a strength of 0.

Sample Input #3

```
4
-3 0
3 0
0 3
0 1
```

Sample Output #3

```
0
```

Explanation for the sample input/output #3

All magical barriers have a strength of 0.

Sample Input #4

```
4
0 0
0 1
1 0
1 1
```

Sample Output #4

```
1
```

Explanation for the sample input/output #4

The strongest magical barrier is either $\langle 1, 4 \rangle$ or $\langle 2, 3 \rangle$, which intersects each other at $(0.5, 0.5)$.

Problem C

Nightmare Brother

Your brother has a string S of length M with indices from 1 to M. You want to know exactly what string S is. To help you, he gives you N hints that might help you to figure out S. Hint i is represented by an integer X_i and a string T_i, indicating that the string T_i appears as a substring of S starting from index X_i of S. All the hints are unique, that is, there are no hints i and j such that $i \neq j$ while $X_i = X_j$ and $T_i = T_j$.

However, your brother is known to be mischievous and tells you that there might be **at most** one false hint among all N hints he has given, but he didn't tell you which.

A string S is a possible solution if and only if there exists a set of at least $N - 1$ hints (that are assumed to be true) where string S is the **only** string consistent with all of the hints in the set.

You would like to find a possible solution. If there is no possible solution, you should output `-1`. If there is more than one possible solution, you should output `-2`.

Input

Input begins with two integers N M ($1 \leq N \leq 100$; $1 \leq M \leq 100$) representing the number of hints and the length of the scary string, respectively. Each of the next N lines contains an integer and a string X_i T_i ($1 \leq X_i, |T_i|$; $X_i + |T_i| - 1 \leq M$) representing hint i. The string T_i consists of only uppercase characters. It is guaranteed that there are no hints i and j such that $i \neq j$ while $X_i = X_j$ and $T_i = T_j$.

Output

If there is exactly one possible solution as explained in the problem description above, then output the string S in a single line. If there is no possible solution, then output `-1` in a single line. If there is more than one possible solution, then output `-2` in a single line.

Sample Input #1

```
3 11
5 JAKARTA
1 ICPC
3 BINUS
```

Sample Output #1

```
ICPCJAKARTA
```

Explanation for the sample input/output #1

The only possible S is `ICPCJAKARTA` assuming hint 3 is false. If the false hint is assumed to be one of the others, then there is no string consistent with the other two hints. Similarly when no hint is assumed false.

Sample Input #2

```
3 9
6 EX
8 AM
1 FINAL
```

Sample Output #2

```
FINALEXAM
```

Explanation for the sample input/output #2

The only possible S is FINALEXAM assuming no hint is false. If any of the hints are assumed to be false, then there is more than one string consistent with the rest of the hints.

Sample Input #3

```
3 8
1 GRAD
5 UAL
6 ATE
```

Sample Output #3

```
-1
```

Explanation for the sample input/output #3

There is no possible solution.

- Assuming no hint is false: There is no string consistent with all the hints.
- Assuming hint 1 is false: There is no string consistent with the other two hints.
- Assuming hint 2 is false or hint 3 is false: There is more than one string consistent with the other two hints.

Sample Input #4

```
3 5
1 BIN
4 US
4 OM
```

Sample Output #4

```
-2
```

Explanation for the sample input/output #4

There are 2 possible solutions: BINOM (assuming hint 2 is false) and BINUS (assuming hint 3 is false).

Problem D
City Hall

You are the mayor of ICPC City. The city has N intersections, numbered from 1 to N, where intersection i has an altitude of H_i. Your house is at intersection S, while the city hall is at intersection T.

There are M two-way roads, numbered from 1 to M, that connect the intersections. Road i directly connects intersections U_i and V_i. Each pair of intersections can only be directly connected by at most one road. The roads connect such that each intersection can be visited from any other intersections by traversing one or more roads.

Every morning, you cycle from your house to the city hall. Suppose that you are traversing a road that directly connects intersections u and v. The energy that you spend to traverse that road is $(H_u - H_v)^2$. The total energy required for a path is the sum of energy that is spent traversing each road in that path.

As a mayor, you are allowed to change the altitude of **at most** one intersection to any non-negative real number. Using this opportunity, you want to minimize the total energy required to cycle from your house to the city hall.

Input

Input begins with 4 integers N M S T ($2 \leq N \leq 100\,000$; $N - 1 \leq M \leq \min(\frac{N(N-1)}{2}, 200\,000)$; $1 \leq S, T \leq N$; $S \neq T$). The next line contains N integers H_i ($0 \leq H_i \leq 100\,000$) representing the altitude of intersection i.

Each of the next M lines contains 2 integers U_i V_i ($1 \leq U_i < V_i \leq N$) representing the intersections directly connected by road i. Each pair of intersections can only be directly connected by at most one road. Furthermore, the roads connect such that each intersection can be visited from any other intersections by traversing one or more roads.

Output

Output a real number in a single line representing the minimum total energy required. Your answer is considered correct if its **absolute error** does not exceed 10^{-6}.

Sample Input #1

```
5 6 1 3
5 100 8 2 10
1 2
2 3
2 5
1 4
4 5
3 5
```

Sample Output #1

```
4.500000
```

Explanation for the sample input/output #1

To get the minimum total required energy, you should change the altitude of intersection 2 to 6.5. Then, cycling route $1 \rightarrow 2 \rightarrow 3$ requires a total energy of $(5 - 6.5)^2 + (8 - 6.5)^2 = 4.5$.

Sample Input #2

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

Sample Output #2

```
3.000000
```

Explanation for the sample input/output #2

To get the minimum total required energy, you can choose either intersection 3 or intersection 4 and change its altitude to 3.

Sample Input #3

```
5 4 1 4
8 8 8 8 100
1 2
2 3
3 4
4 5
```

Sample Output #3

```
0.000000
```

Explanation for the sample input/output #3

The cycling route $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ requires a total energy of 0 as all its intersections are at the same height. Changing any intersection's height will not decrease the total energy required.

Problem E
Substring Sort

In this problem, all strings are one-based indexed. Let s_i be the i^{th} character of a string s. Let $s_{l..r}$ be a substring of s with the characters $s_l s_{l+1} \cdots s_r$.

You are given three strings each of length N: A, B, and C. You are asked to simulate Q queries according to the given order.

For each query, you are given two integers l and r as parameters, and must performs the following procedures:

1. Copy substrings $A_{l..r}$, $B_{l..r}$, and $C_{l..r}$. Let x, y and z be the copied substrings.
2. Sort $[x, y, z]$ in lexicographical order. Let $[x', y', z']$ be the sorted results.
3. Replace substring $A_{l..r}$ with x', substring $B_{l..r}$ with y' and substring $C_{l..r}$ with z' respectively.

Determine the value of A, B, and C after all queries.

Input

Input begins with two integers N Q ($1 \leq N \leq 100\,000$; $1 \leq Q \leq 100\,000$) representing the length of the given strings and the number of queries. Each of the next 3 lines contains a string of length N. The first, second, and third lines contain A, B, and C respectively. The strings consist of lowercase characters. Each of the next Q lines contains two integers l r ($1 \leq l \leq r \leq N$) representing the parameters of each query.

Output

The output consists of 3 lines. In each line, output the final value of A, B and C after all queries in that order.

Sample Input #1

```
5 2
icpca
siaja
karta
2 4
1 5
```

Sample Output #1

```
iarta
kiaja
scpca
```

Explanation for the sample input/output #1

In the first query, the value of x, y, and z are `cpc`, `iaj`, `art` respectively. After sorting those strings, the value of x', y', and z' becomes `art`, `cpc`, `iaj`. At the end of first query, the value of A, B, and C are `iarta`, `scpca` and `kiaja`.

During the second query, the value of x, y, and z are `iarta`, `scpca`, `kiaja` respectively. After sorting those strings, the value of x', y', and z' becomes `iarta`, `scpca`, `kiaja`. At the end of second query, the value of A, B, and C are `iarta`, `kiaja` and `scpca`.

Therefore the final value of A, B, C are `iarta`, `kiaja` and `scpca` respectively.

Sample Input #2

```
6 6
aabbcc
bcacab
cbcaba
1 1
2 2
3 3
4 4
5 5
6 6
```

Sample Output #2

```
aaaaaa
bbbbbb
cccccc
```

Sample Input #3

```
3 1
aba
aab
aac
1 3
```

Sample Output #3

```
aab
aac
aba
```

Problem F

Doubled GCD

There are N cards in a deck, numbered from 1 to N, where card i has a positive integer A_i written on it.

You are to perform $N - 1$ moves with the cards. In each move, you select two cards of your choice from the deck. Let x and y be the integers written on the selected cards, respectively. Remove both selected cards, and insert a new card into the deck with $2 \cdot gcd(x, y)$ written on it, where $gcd(x, y)$ is the greatest common divisor of x and y. Note that with this one move, there will be one fewer card in the deck (as you remove two cards and insert one new card).

After all $N-1$ moves have been performed, there will be exactly one card remaining. Your goal is to maximize the integer written on the last card; output this integer.

Input

Input begins with an integer N ($2 \leq N \leq 100\,000$) representing the number of cards. The next line contains N integers A_i ($1 \leq A_i \leq 10^9$) representing the number written on card i.

Output

Output an integer in a single line representing the maximum possible integer written on the last card.

Sample Input #1

```
3
2 4 6
```

Sample Output #1

```
8
```

Explanation for the sample input/output #1

To get the maximum possible integer on the last card, you have to select card 1 and card 3 on the first move with $x = 2$ and $y = 6$. Remove both selected cards, and insert a new card with $2 \cdot gcd(2, 6) = 4$ written on it. For the second move, there are two cards remaining with an integer 4 written on each card. Select those cards with $x = 4$ and $y = 4$. Remove both selected cards, and insert a new card with $2 \cdot gcd(4, 4) = 8$ written on it. The last card has an integer 8 written on it, and it is the maximum possible integer in this example.

Sample Input #2

```
3
3 5 7
```

Sample Output #2

```
2
```

Explanation for the sample input/output #2

Regardless of your choice in each move, the answer will always be 2.

Sample Input #3

```
4
9 9 9 9
```

Sample Output #3

```
36
```

Sample Input #4

```
5
10 100 1000 10000 100000
```

Sample Output #4

```
160
```

Problem G
The Only Mode

You are given an array of integers A of size N (indexed from 1 to N) where A_i is either 0, 1, 2, or 3.

A subarray $\langle l, r \rangle$ of A is defined as $[A_l, A_{l+1}, \cdots, A_r]$, and its size is $r - l + 1$.

A value x is the *only mode* of a subarray $\langle l, r \rangle$ if and only if x appears **strictly** more often than other values in subarray $\langle l, r \rangle$.

Your task in this problem is to find, for each $x \in \{0, 1, 2, 3\}$, the size of the longest subarray of A such that x is the only mode of that subarray, or determine if x cannot be the only mode in any subarray.

Input

Input begins with an integer N ($1 \leq N \leq 100\,000$) representing the size of array A. The next line contains N integers A_i ($A_i \in \{0, 1, 2, 3\}$).

Output

Output four space-separated integers in a single line. Each integer represents the answer where x is 0, 1, 2, and 3, respectively. For each value of x, if there exists a subarray such that x is the only mode in that subarray, then output the size of the longest subarray; otherwise, output 0.

Sample Input #1

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

Sample Output #1

```
4 1 5 3
```

Explanation for the sample input/output #1

- The longest subarray such that 0 is the only mode is $\langle 3, 6 \rangle$ of length 4, i.e. $[2, 0, 3, 0]$.
- The longest subarray such that 1 is the only mode is $\langle 1, 1 \rangle$ of length 1, i.e. $[1]$.
- The longest subarray such that 2 is the only mode is $\langle 1, 5 \rangle$ of length 5, i.e. $[1, 2, 2, 0, 3]$.
- The longest subarray such that 3 is the only mode is $\langle 5, 7 \rangle$ of length 3, i.e. $[3, 0, 3]$.

Sample Input #2

```
12
2 0 1 0 2 1 1 0 2 3 3 3
```

Sample Output #2

```
4 9 1 9
```

Explanation for the sample input/output #2

- The longest subarray such that 0 is the only mode is $\langle 1, 4\rangle$ or $\langle 2, 5\rangle$.
- The longest subarray such that 1 is the only mode is $\langle 3, 11\rangle$.
- The longest subarray such that 2 is the only mode is $\langle 1, 1\rangle$, $\langle 5, 5\rangle$, or $\langle 9, 9\rangle$.
- The longest subarray such that 3 is the only mode is $\langle 4, 12\rangle$.

Sample Input #3

```
2
0 2
```

Sample Output #3

```
1 0 1 0
```

Explanation for the sample input/output #3

The longest subarray such that 0 or 2 is the only mode contains only a single element by itself; on the other hand, there is no subarray such that 1 or 3 is the only mode.

Sample Input #4

```
12
3 0 2 2 1 0 2 1 3 3 2 3
```

Sample Output #4

```
1 5 11 8
```

Problem H
Grid Game

Given a grid A of size $N \times M$. Each row is numbered from 1 to N, and each column is numbered from 1 to M. The cell at row r and column c is denoted as (r, c).

Cell (r, c) contains an integer $A_{r,c}$, which can be either -1 or a non-negative integer. If $A_{r,c} = -1$, that means cell (r, c) is *impassable*. Otherwise, cell (r, c) is *passable*.

Two players will alternately take turns playing on this grid. In one turn, a player will do the following.

1. Choose a cell with **positive** integer on it, and the player starts standing on that cell. Let x be the integer at this starting cell.
2. Choose a **non-negative** integer y such that $y < x$.
3. Suppose that the player is standing on cell (r, c). Update the value of $A_{r,c}$ to $A_{r,c} \oplus x \oplus y$, where $\oplus$ is the bitwise operator XOR.
4. If either cell $(r + 1, c)$ or cell $(r, c + 1)$ is passable, the player must move to either passable cell of the player's choosing. Then, repeat from step 3.
5. If the player is no longer able to move, the player will step outside of the grid and end his turn.

A player who is unable to play on his turn (i.e. no positive integer on his turn) loses the game, and the opposing player wins the game.

If both players play optimally, determine who will win the game.

Input

Input begins with two integers N M ($1 \leq N, M \leq 500$) representing the size of grid A. Each of the next N lines contains M integers $A_{r,c}$ ($0 \leq A_{r,c} \leq 10^9$ or $A_{r,c} = -1$) representing the integer contained in cell (r, c).

Output

If the first player win the game, output `first` in a single line. Otherwise, output `second` in a single line.

Sample Input #1

```
5 6
0 0 0 0 0 0
0 3 3 0 0 0
0 0 3 -1 0 0
0 0 3 3 3 3
0 0 -1 -1 -1 -1
```

Sample Output #1

```
first
```

Explanation for the sample input/output #1

The first player can start his turn from $(2, 2)$ and choose $y = 0$. Then, he can keep following the cells with integer 3 and update those cells to $3 \oplus 3 \oplus 0 = 0$. After the first turn, there is no positive integer on the grids, thus second player is unable to play.

Sample Input #2

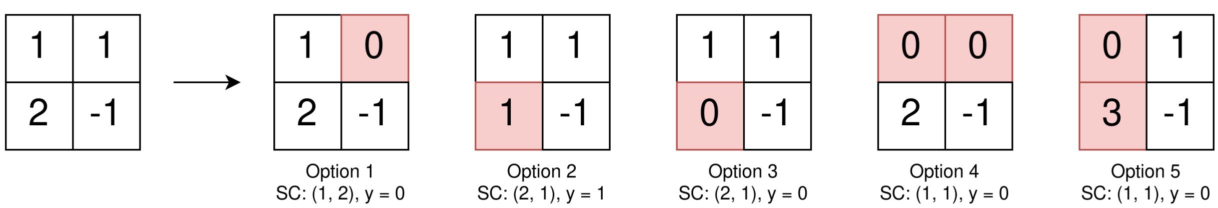
```
2 2
1 1
2 -1
```

Sample Output #2

```
first
```

Explanation for the sample input/output #2

There are 5 options that can be made by the first player during the first turn, as shown in the following illustration. Each option has different starting cell (abbreviated as $\texttt{SC}$), the value of y, and the path taken by the first player. The taken paths are indicated by the red shaded cells.



The optimal moves for the first player to guarantee his win is either option 1 or 2.

Sample Input #3

```
1 1
-1
```

Sample Output #3

```
second
```

Explanation for the sample input/output #3

The initial grid has no positive integer cell. The second player wins the game by default.

Problem I
Contingency Plan

You are working as a manager in The ICPC Company. In the company building, there are N computers, numbered from 1 to N. There are $N - 1$ cables, numbered from 1 to $N - 1$, that connect all the computers into a single network. Cable i connects computer U_i and V_i.

Through your research, there are $N - 1$ levels of disasters, numbered from 1 to $N - 1$, that might happen in the future. In disaster level x, all cables i such that $1 \le i \le x$ are damaged. Damaged cables cannot be used for a connection.

As a manager, you want to create a contingency plan. In your contingency plan, there should be $N - 1$ backup cables, numbered from 1 to $N - 1$. If an existing cable i is damaged, then backup cable i will be deployed to connect computer A_i and B_i. If an existing cable i is not damaged, then backup cable i is not deployed and is not used for a connection.

For each disaster level, the backup cables, together with the undamaged cables, must keep all the computers connected in a single network. Furthermore, for practical reasons, if a cable that connects computers u and v exists, then there should not be any backup cable that connects computers u and v in your contingency plan.

Create a contingency plan that satisfies all the requirements, or determine if such a plan is impossible to create. If several contingency plans exist, choose any of them.

Input

Input begins with an integer N ($2 \le N \le 100\,000$) representing the number of computers. Each of the next $N - 1$ lines contains 2 integers U_i V_i ($1 \le U_i, V_i \le N$) representing cable i. All the cables connect all the computers into a single network.

Output

If a contingency plan is possible to create, then the output consists of $N - 1$ lines, representing your contingency plan that satisfies all the requirements. Each line contains 2 integers A_i B_i ($1 \le A_i, B_i \le N$) representing backup cable i. If several contingency plans exist, output any of them.

If a contingency plan is impossible to create, then output -1 in a single line.

Sample Input #1

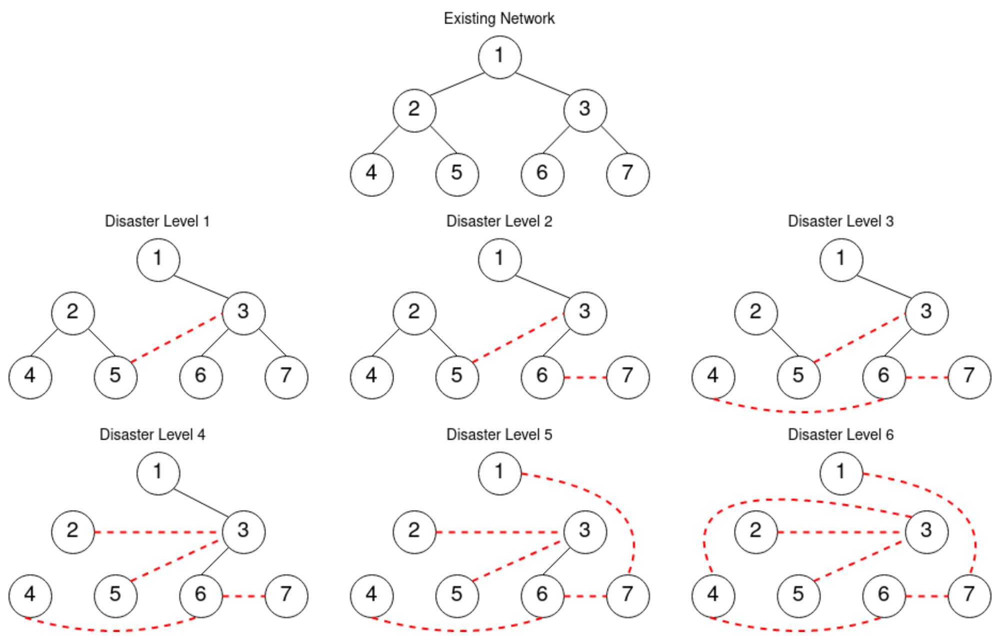
```
7
1 2
3 7
2 4
2 5
1 3
3 6
```

Sample Output #1

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

Explanation for the sample input/output #1

The following is an illustration for this sample. The circles represent the computers, while the black line and red dashed lines represent existing cables and backup cables, respectively. Damaged cables are not shown in the illustration.



Sample Input #2

```
3
1 2
2 3
```

Sample Output #2

```
-1
```

Explanation for the sample input/output #2

There should be 2 backup cables in the contingency plan. You can only have 1 backup cable, which will connect computers 1 and 3. The remaining pairs are already connected by the current cables.

Problem J
Sharing Bread

There are N toasters, numbered from 1 to N, from left to right. Initially, each toaster has a single piece of bread in it. There are M people, numbered from 1 to M, who are one by one looking for bread among the toasters, starting from person 1, person 2, and so on.

Person i starts looking from toaster a_i ($1 \le a_i \le N$) and keeps going right until they found a toaster with a piece of bread in it. In other words, person i is looking for the smallest j such that $a_i \le j \le N$ and toaster j contains bread. If such a toaster exists, then person i will take the bread from that toaster and leave; the toaster becomes empty afterward. If such a toaster does not exist, then person i will leave empty-handed.

A starting sequence $(a_1, a_2, \cdots, a_M)$ is *fair* if person i starts looking from toaster a_i and does not leave empty-handed, for all $1 \le i \le M$. Out of all N^M possible starting sequences, determine how many of them are fair modulo $998\,244\,353$.

Input

Input consists of two integers N M ($1 \le M \le N \le 200\,000$) in a single line representing the number of toasters and the number of people, respectively.

Output

Output an integer in a single line representing the number of fair starting sequence modulo $998\,244\,353$.

Sample Input #1

```
4 3
```

Sample Output #1

```
50
```

Explanation for the sample input/output #1

One of the possible fair starting sequences is $(4, 2, 2)$. First, person 1 starts looking from toaster 4 and takes the bread from toaster 4. Then, person 2 starts looking from toaster 2 and takes the bread from toaster 2. Finally, person 3 will start looking from toaster 2, which is currently empty. Person 3 moves to toaster 3 and takes the bread from toaster 3. Since each person gets a piece of bread, the starting sequence $(4, 2, 2)$ is fair.

Another example of fair starting sequences are $(1, 1, 1)$, $(1, 1, 2)$, $(2, 3, 4)$, and $(2, 2, 2)$. Some of the possible starting sequences that are not fair are $(3, 3, 3)$, $(3, 4, 3)$, $(4, 4, 1)$, and $(4, 4, 4)$.

Sample Input #2

```
10 1
```

Sample Output #2

```
10
```

Explanation for the sample input/output #2

All starting sequences are fair.

Sample Input #3

```
2 2
```

Sample Output #3

```
3
```

Explanation for the sample input/output #3

The only starting sequence that is **not** fair is $(2, 2)$. Person 1 starts looking from toaster 2 and takes the bread from toaster 2. Then, person 2 starts looking from toaster 2, which is currently empty. Since there is no more toaster to the right of toaster 2, person 2 will leave empty-handed.

Problem K
Short Function

Last week, your algorithm course's lecturer gave you an assignment to determine the output of a given pseudocode function. Even though the assignment contains only a single problem, the lecturer warned you not to underestimate it and suggests you spend more time doing it.

The following is the snapshot of the assignment that you need to finish before the deadline.

> Given an array of positive integers `A[]` of length `N` (indexed from `0` to `N-1`), an integer `K`, and the following pseudocode function. Your task in this problem is to determine the output of the following function from the given input.
>
> ```
> SomeFunction(A[0..N-1], N, K):
>     B[0..N-1] = A[0..N-1]
>     for i = 0 to K-1:
>         A[0..N-1] = B[0..N-1]
>         for j = 0 to N-1:
>             B[j] = A[j] × A[(j + 2^i) mod N]
>     return B[0..N-1]
> ```
>
> What is the output of the function (i.e. what are the values for `B[0..N-1]`)? Please ask your teaching assistant for the input `A[]`, `N`, and `K`.
>
> IMPORTANT: As the return value for `B[0..N-1]` can be very large, it can be very troublesome to verify, so you must modulo each element of `B[0..N-1]` by $998\,244\,353$.

As the problem looks short and easy, you decided to leave the assignment to the last minute before the submission deadline. You managed to get the required input (the array A, integer N, and integer K) from the teaching assistant, but you quickly regret your lazy decision after implementing the function pseudocode. Apparently, a direct implementation of the function might need hours to run.

Now you need to calm down and figure out the output of the function given such input before the deadline.

Input

Input begins with two integers N K ($1 \leq N \leq 100\,000$; $1 \leq K \leq 10^9$) representing the size of input array A and the given integer, respectively. The next line contains N integers A_i ($1 \leq A_i < 998\,244\,353$) representing the elements of array A.

Output

Output N integers in a single line, each separated by a single space, representing the output of the function (i.e. the array `B[]`). Modulo each element in `B[]` by $998\,244\,353$. See sample output for clarity.

Sample Input #1

```
5 2
1 2 3 4 5
```

Sample Output #1

```
24 120 60 40 30
```

Sample Input #2

```
8 3
12 5 16 14 10 6 9 2
```

Sample Output #2

```
14515200 14515200 14515200 14515200 14515200 14515200 14515200 14515200
```

Sample Input #3

```
6 10
3 7 8 2 9 5
```

Sample Output #3

```
56347321 169041963 833775940 811788154 844769833 639990479
```

Sample Input #4

```
2 100
1 2
```

Sample Output #4

```
917380677 917380677
```

Problem L

Increase the Toll Fees

The ICPC Kingdom is a big kingdom with N cities, numbered from 1 to N. The charm of the ICPC Kingdom lies in its beautiful sceneries in the kingdom. To promote those sceneries, the King of the ICPC Kingdom decided to make M toll roads, numbered from 1 to M, near the scenic views for the tourists to enjoy. To use toll road i that connects city U_i to V_i bidirectionally, one must pay W_i. It is possible to travel from one city to any other city using these toll roads.

Although those toll roads have been built and can be used, they still do not attract tourists. The King decided to make a promotion, where one can **pay in advance** for the toll roads they want to use and can use it **multiple times** as long as they do not leave the ICPC Kingdom.

This idea can finally attract tourists, but there's a strange behaviour happening. All of the tourists only pay for the set of toll roads that gives the minimum total pay which allows them to travel from one city to any other city regardless of the distance. Interestingly, such a set of toll roads is **unique** under current toll pricing. This strange behaviour does not fully expose the kingdom's scenery to the tourists.

To promote more scenery, the King decided to increase the price of some toll roads. If toll road i is used by the tourists' strange behaviour before the toll price increase, then after the price increase, the King **must ensure** toll road i is **not used** by the tourists' strange behaviour. For stability, the King also wants the total price increase across all toll roads to be as small as possible.

The King asked you to calculate what is the minimum total increase to fulfill the King's plan or report that it is impossible to do so.

Input

Input begins with two integers N M ($2 \le N \le 100\,000$; $N - 1 \le M \le 200\,000$) representing the number of cities and toll roads. Each of the next M lines contains 3 integers U_i V_i W_i ($1 \le U_i < V_i \le N$; $1 \le W_i \le 10^9$) representing toll i that connects city U_i and V_i with price W_i. There exists a unique set of toll roads that allow travel between any two cities with minimum total pay before the price increase.

Output

If the King's plan is possible to achieve, then output an integer representing the minimum total increase to fulfill the King's plan. Otherwise, output -1 in a single line.

Sample Input #1

```
4 6
1 2 2
1 3 5
1 4 5
2 3 3
2 4 5
```

```
3 4 4
```

Sample Output #1

```
9
```

Before the price increase, the tourists will choose toll roads 1, 3, and 6 to travel. By increasing the price of toll roads 1, 3, and 6, to 6, the tourists will use toll roads 2, 4, and 5 to travel. Total increase toll roads is $(6 - 2) + (6 - 3) + (6 - 4) = 6$.

Sample Input #2

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

Sample Output #2

```
-1
```

Explanation for the sample input/output #2

Before the price increase, the tourists will choose toll roads 1 and 2. No matter how many the price increase, the tourists will always choose at least one of those two toll roads.

Sample Input #3

```
5 10
1 2 14
1 3 14
1 4 9
1 5 15
2 3 8
2 3 10
2 4 13
3 4 8
4 5 10
4 5 15
```

Sample Output #3

```
21
```

Problem M
Game Show

You are hosting a game show. In your game show, there is a circular disk divided into N regions, numbered from 1 to N in clockwise order. For each region i ($1 \leq i \leq N - 1$), region $i + 1$ is located to the next of region i, and region 1 is located to the next of region N.

There are Q independent rounds. In each round, the player starts from region S and the target is at region T. For each i such that $1 \leq i \leq N$, the player can move from region i to region $i + 1$ (or to region 1 if $i = N$) with a penalty of A_i. Similarly, the player can move from region $i + 1$ (or from region 1 if $i = N$) to region i with a penalty of B_i. Note that the penalty can be negative.

The goal of each round is to find the minimum total penalty required to reach the target. However, you noticed that it is possible for the player to abuse the game to reach the target with a penalty of $-\infty$. Such round is called *flawed*.

For each round, determine if the round is flawed or not. If the round is not flawed, determine the minimum penalty to reach the target.

Input

Input begins with two integers N Q ($3 \leq N \leq 200\,000$; $1 \leq Q \leq 200\,000$) representing the number of regions and the number of rounds, respectively.

The next line contains N integers A_i ($-10^9 \leq A_i \leq 10^9$) representing the penalty to move from region i to region $i + 1$, or to region 1 if $i = N$. The next line contains N integers B_i ($-10^9 \leq B_i \leq 10^9$) representing the penalty to move from region $i + 1$, or from region 1 if $i = N$, to region i.

Each of the next Q lines contains two integers S T ($1 \leq S, T \leq N$) representing the start region and target region of each round, respectively.

Output

For each round, if the round is flawed, then output `flawed` in a single line. Otherwise, output an integer in a single line, representing the minimum penalty to reach the target.

Sample Input #1

```
4 4
2 3 -4 3
1 2 7 -1
1 3
3 1
1 4
1 1
```

Sample Output #1

```
5
-1
-1
0
```

Explanation for the sample input/output #1

In round 1, the path $1 \to 2 \to 3$ has a penalty of $2 + 3 = 5$.

In round 2, the path $3 \to 4 \to 1$ has a penalty of $(-4) + 3 = -1$. This path has lesser penalty than path $3 \to 2 \to 1$, which has a penalty of $2 + 1 = 3$.

In round 3, the path $1 \to 4$ has a penalty of -1.

Sample Input #2

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

Sample Output #2

```
flawed
flawed
flawed
```

Explanation for the sample input/output #2

For all rounds, the player can go to region 2, then repeatedly travel back and forth in regions 2 and 3 to reduce the penalty by 1 infinitely many times.

Sample Input #3

```
6 2
-6 8 -3 5 -9 4
9 -2 8 -4 12 -1
2 6
3 3
```

Sample Output #3

```
flawed
flawed
```