Problem A
Party Medley

ICPC University has N students, numbered from 1 to N, in its Competitive Programming club. Student i has a rating of R_i representing their estimated skill in competitive problem-solving.

Contest season is coming and Morgan, the coach of the Competitive Programming club, would like to send at most one good team to a particular contest due to their limited budget. A team consists of exactly 3 different students. Suppose that a team consists of student i, j, and k. Their *team rating* is $A_i + A_j + A_k$, and their *rating difference* is $\max(A_i, A_j, A_k) - \min(A_i, A_j, A_k)$.

Morgan believes that a team is *balanced* if their rating difference is no more than a threshold of M. Additionally, he also would like the team rating to be as large as possible while being a balanced team as well.

Morgan asks you to compute two values. The first value is the number of different balanced team configurations that can be made. The second value is the largest team rating of a balanced team that can be made.

Two team configurations are different if and only if there is at least one different student between those team configurations.

Input

Input begins with two integers N M ($3 \leq N \leq 200$; $0 \leq M \leq 4000$) representing the number of students and the threshold for rating difference, respectively. The next line contains N integers A_i ($0 \leq A_i \leq 4000$) representing the rating of student i.

Output

If there is at least one balanced team configuration, then output two space-separated integers in a single line representing the number of different balanced team configurations and the largest team rating of any balanced team, respectively.

If there is no balanced team configuration, then output -1 in a single line.

Sample Input #1

```
5 150
1400 1425 1250 4000 1300
```

Sample Output #1

```
2 4125
```

Explanation for the sample input/output #1

An example of a balanced team configuration is the team consisting of student 1, 3, and 5. Their team rating is $1400 + 1250 + 1300 = 3950$. Their rating difference is $1400 - 1250 = 150$, which is no more than 150.

The other balanced team configuration is the team consisting of students 1, 2, and 5. Their rating difference is 125 with a team rating of 4125, which is the highest team rating among all balanced team configurations that can be made.

Sample Input #2

```
4 100
2000 1900 1800 2100
```

Sample Output #2

```
-1
```

Explanation for the sample input/output #2

Any team configuration has a rating difference of at least 200, which is more than the given threshold.

Sample Input #3

```
8 4000
100 200 300 400 500 600 700 800
```

Sample Output #3

```
56 2100
```

Explanation for the sample input/output #3

Any team configuration in this example is a balanced team, while the team consisting of students 6, 7, and 8 has the largest team rating of $600 + 700 + 800 = 2100$

Sample Input #4

```
8 0
10 10 10 20 20 20 30 30
```

Sample Output #4

```
2 60
```

Explanation for the sample input/output #4

There are only 2 possible balanced team configurations: A team with students 1, 2, and 3 with a total team rating of $10 + 10 + 10 = 30$, and a team with students 4, 5, and 6 with a total team rating of $20 + 20 + 20 = 60$. The latter has the largest team rating.

Problem B
Cooking Steaks

Morgan is a chef in a steak house. In his steak house, a steak can have N level of doneness, numbered from 1 to N. Currently, Morgan has A_i steaks of doneness level i ready in his steak house.

There are B_i orders of steaks with doneness level i that need to be fulfilled. Morgan can cook the steaks in order to match the doneness level. For each $1 \leq i < N$, it takes Morgan T_i seconds to cook a steak from doneness level i to $i + 1$. Note that Morgan can only cook one steak at a time.

Morgan asks for your help to find the minimum total time to fulfil all orders, or tell him that the orders are impossible to fulfil.

Input

Input begins with an integer N ($2 \leq N \leq 100\,000$). The next line contains $N - 1$ integers T_i ($1 \leq T_i \leq 1000$) representing the time required to cook a steak of doneness level i to $i+1$. The next line contains N integers A_i ($0 \leq A_i \leq 1000$) representing the number of steaks with doneness level i. The next line contains N integers B_i ($0 \leq B_i \leq 1000$) representing the number of orders for a steak with doneness level i.

Output

If all orders can be fulfilled, then output an integer in a single line representing the minimum total time to fulfil all orders. Otherwise, output `-1` in a single line.

Sample Input #1

```
3
1 2
2 2 3
0 1 5
```

Sample Output #1

```
5
```

Explanation for the sample input/output #1

First, Morgan can cook both steaks with doneness level 2 to level 3 in 2 seconds each. Then, Morgan can cook one steak with doneness level 1 to level 2 in 1 second. Now, Morgan has 1 steak of doneness level 1, 1 steak of doneness level 2, and 5 steaks of doneness level 3. It is enough to fulfil all orders. There is no other way to fulfil all orders in less than 5 seconds.

Sample Input #2

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

Sample Output #2

```
0
```

Explanation for the sample input/output #2

The steaks ready in his steak house can fulfil all orders without any further cooking.

Sample Input #3

```
3
1 2
2 2 3
5 0 0
```

Sample Output #3

```
-1
```

Explanation for the sample input/output #3

It is impossible to have 5 steaks of doneness level 1.

Problem C

Powers of Two

Adrian has learned addition and subtraction from Morgan and is now ready to learn a new concept, the powers of two. Powers of two are integers in the form of 2^x, where $x \geq 0$. Some examples of powers of two are $1, 2, 4, 8, \ldots$.

To ensure Adrian understands this new concept, Morgan prepares a challenge for him. At first, Adrian is given an integer $N = 0$. Then, Morgan will give him Q queries. Each query can be one of the following types:

- + x, which will add the value of N by 2^x, or
- - x, which will subtract the value of N by 2^x.

Adrian is instructed to clap his hands whenever N becomes 0 after each query.

Adrian finds this challenge is very hard to follow. He asks you whether he should clap or not after each query.

Input

Input begins with an integer Q ($1 \leq Q \leq 200\,000$) representing the number of queries. Each of the next Q lines contains a character and an integer T x ($T \in \{$ +, - $\}$; $0 \leq x \leq 200\,000$) representing the query.

Output

After each query, output `YES` in a single line if the value of N becomes 0, or output `NO` otherwise.

Sample Input #1

```
6
+ 3
+ 3
- 4
- 6
+ 7
- 6
```

Sample Output #1

```
NO
NO
YES
NO
```

```
NO
YES
```

Explanation for the sample input/output #1

The value of N after each query is 8, 16, 0, -64, 64, and 0. Therefore, Adrian should clap after query 3 and 6.

Sample Input #2

```
13
+ 13324
+ 5773
- 5772
+ 13324
+ 0
- 5772
- 13325
- 0
+ 0
+ 0
- 200000
- 1
+ 200000
```

Sample Output #2

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

Problem D
Robot Upgrades

The Kingdom of ICPC is being attacked by evil balloons! Fortunately, Morgan the robot is ready to defend the kingdom. In order to strengthen his power, there are N parts, numbered from 1 to N, that can be upgraded. Each part can be upgraded 0 to M times (inclusive).

In order to save resources, there are M restrictions, numbered from 1 to M, in upgrading Morgan. For restriction i, the number of parts that are upgraded at least i times should not exceed A_i.

While planning on what upgrade should be applied to Morgan, Adrian wonders how many different upgrade configurations that satisfy all of the given restrictions. Two configurations are different if and only if there exists at least one part with a different number of upgrades applied to that part. Since the answer can be large, find the answer modulo $998\,244\,353$.

Input

Input begins with two integers N M ($1 \leq N \leq 100\,000$; $1 \leq M \leq 10$) representing the number of parts and the number of restrictions, respectively. The next line contains M integers A_i ($1 \leq A_i \leq N$) representing the given restrictions. The integers in A are given in non-increasing order, i.e. $A_1 \geq A_2 \geq \ldots \geq A_M$.

Output

Output an integer in a single line, representing the number of different upgrade configurations that satisfy all of the given restrictions modulo $998\,244\,353$.

Sample Input #1

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

Sample Output #1

```
64
```

Explanation for the sample input/output #1

Denote $(u_1, u_2, \ldots, u_N)$ as an upgrade configuration such that part i is upgraded u_i times for all $1 \leq i \leq N$.

Some configurations that satisfy all restrictions are $(0,0,0)$, $(0,0,5)$, $(2,2,0)$, $(2,0,4)$, and $(0,1,0)$. Some configurations that do not satisfy all restrictions are $(1,1,1)$, $(3,0,3)$, $(5,5,5)$, $(0,5,3)$, and $(2,1,4)$.

Sample Input #2

```
1 2
1 1
```

Sample Output #2

```
3
```

Explanation for sample input/output #2

All the upgrade configurations are to upgrade the only part 0, 1, or 2 times.

Sample Input #3

```
16 8
16 16 8 4 4 2 1 1
```

Sample Output #3

```
720246211
```

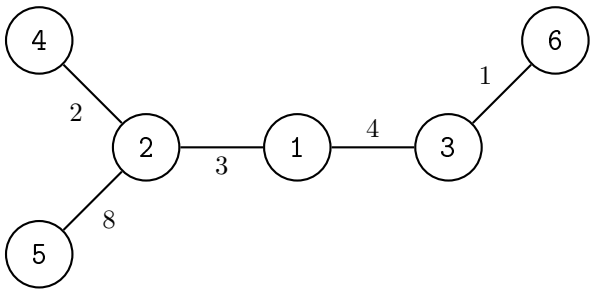
Problem E
Walking Around

You are given a weighted tree with N vertices, numbered from 1 to N. The edges are numbered from 1 to $N-1$, where edge i connects two vertices U_i and V_i with a weight of a non-negative integer W_i.

A path in the tree is defined as a sequence of unique vertices $(u_0, u_1, \ldots, u_m)$ for some $m \geq 1$ such that each pair of adjacent vertices, (u_j, u_{j+1}) for all $0 \leq j < m$, is connected by an edge in the tree. Define the *score* of a path $(u_0, u_1, \ldots u_m)$ as the bitwise XOR of the weight of all edges in the path, i.e. $\mathrm{XOR}(w_0, w_1, \ldots, w_{m-1})$ where w_j is the weight of the edge that connects u_j and u_{j+1} (for all $0 \leq j < m$).

Your task is to find the minimum and the maximum score of any path that can be obtained from the given tree.

For example, the minimum and the maximum score of any path in the following tree are path $(4, 2, 1)$ with a score of $\mathrm{XOR}(2, 3) = 1$, and path $(5, 2, 1, 3)$ with a score of $\mathrm{XOR}(8, 3, 4) = 15$, respectively.



Input

Input begins with an integer N ($2 \leq N \leq 100\,000$) representing the number of vertices in the given tree. Each of the next $N-1$ lines contains three integers U_i V_i W_i ($1 \leq U_i < V_i \leq N$; $0 \leq W_i \leq 10^9$) representing edge i.

Output

Output two space-separared integers in a single line, representing the minimum and the maximum score of any path that can be obtained from the given tree in that order.

Sample Input #1

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

Sample Output #1

```
1 15
```

Explanation for the sample input/output #1

The illustration in the description section represents this example.

Sample Input #2

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

Sample Output #2

```
1 7
```

Sample Input #3

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

Sample Output #3

```
0 14
```

Problem F
Stock Market

Adrian owns a stock that he previously purchased, and wants to sell that stock. Currently, at day 0, the price of the stock is P_0. As a robot, Morgan can predict the future. Morgan tells Adrian that the price changes will repeat every N days.

Formally, suppose that the price change from day i to day $i+1$ for $0 \le i \le N-1$ is D_i. The price change from day i to $i+1$ for $i \ge N$ is $D_i = D_{i \bmod N}$. The price of the stock at day i for $i > 0$ is $P_i = P_{i-1} + D_{i-1}$. It is possible for a price to be negative.

Moreover, Morgan also knows that the price is on a downward trend. That is, the sum of all D_i is negative.

The following table is the stock price of each day if $N = 6$, $P_0 = 20$, and $D_{0..5} = [4, -6, -1, 4, -9, -2]$.

Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	⋯
Price	20	24	18	17	21	12	10	14	8	7	11	2	0	4	−2	−3	1	−8	⋯

Adrian can only sell the stock when the price is at least X, the price when he purchased the stock, to avoid any losses. As a thrill seeker, Adrian also would like to sell his stock at the lowest price possible while still being at least X.

Help Adrian to determine the lowest price of the stock that is not lower than X, or tell him if it is impossible. Note that Adrian can sell his stock at day 0, if $P_0 \ge X$.

Input

Input begins with three integers N P_0 X ($1 \le N \le 100\,000$; $1 \le P_0, X \le 10^9$) representing the number of days in a cycle, the price at day 0, and the price when Adrian purchased the stock, respectively. The next line contains N integers D_i ($-10^9 \le D_i \le 10^9$) representing the price changes that repeat every N days. It is guaranteed that the sum of all D_i is negative.

Output

If a price not lower than X exists, output an integer in a single line representing the lowest price of the stock that is not lower than X. Otherwise, output `-1` in a single line.

Sample Input #1

```
6 20 5
4 -6 -1 4 -9 -2
```

Sample Output #1

```
7
```

Explanation for the sample input/output #1

The table in the description represents this example. The lowest price of the stock that is not lower than 5 is $P_9 = 7$.

Sample Input #2

```
6 20 1
4 -6 -1 4 -9 -2
```

Sample Output #2

```
1
```

Explanation for the sample input/output #2

The lowest price of the stock that is not lower than 1 is $P_{16} = 1$.

Sample Input #3

```
1 3 4
-1
```

Sample Output #3

```
-1
```

Explanation for the sample input/output #3

The current price, P_0, is 3 and for any subsequent day, the price will never go up. Thus, it is impossible for the price to reach at least $X = 4$.

Problem G
Plus or Times

Adrian is playing a game. When the game starts, Adrian will be given P points as his initial points. The game consists of N rounds, numbered from 1 to N. During round i, Adrian has two options. Each option can be one of the following types:

- `+` c ($-1000 \leq c \leq 1000$) which will add his current points by c, or
- `x` c ($-2 \leq c \leq 2$) which will multiply his current points by c.

Adrian wants to maximize his points at the end of the game. Help Adrian to determine the maximum points he can achieve after completing all N rounds!

Input

Input begins with two integers N P ($1 \leq N \leq 50$; $-1000 \leq P \leq 1000$) representing the number of rounds and the initial points during the game, respectively. Each of the next N lines contains the two options in each round separated by a space. Each option is given in the format T c ($T \in$ { `+`, `x` }; $-1000 \leq c \leq 1000$ if $T =$ `+`, or $-2 \leq c \leq 2$ if $T =$ `x`).

Output

Output an integer in a single line representing the maximum points Adrian can achieve at the end of the game.

Sample Input #1

```
3 123
+ 100 x 2
+ -100 x -2
+ 0 + 0
```

Sample Output #1

```
146
```

Explanation for the sample input/output #1

Adrian can choose the second option in round 1, first option in round 2, and any option in round 3.

Sample Input #2

```
3 123
+ 100 x 2
+ -100 x -2
x 0 x 0
```

Sample Output #2

```
0
```

Explanation for the sample input/output #2

Adrian will always achieve 0 points regardless of his decision in each round, because round 3 will multiply his points by 0.

Problem H
Sorting Machine

Adrian knows that Morgan the robot is capable of sorting texts, but he is uncertain about Morgan's efficiency in doing the task. Adrian decides to give Morgan a test on his efficiency.

First, Adrian gives Morgan a list of N equal-length texts, numbered from 1 to N. Each text is a string S_i that contains M characters, indexed from 1 to M. S_{ij} represents character j in string S_i.

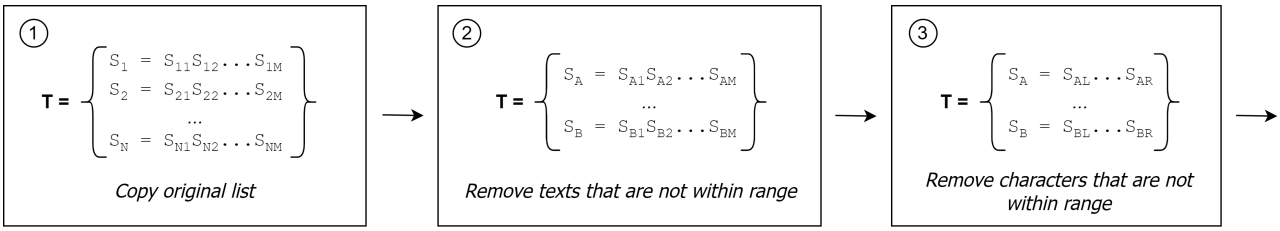
Adrian will give Morgan Q tasks. Each task is represented by a tuple $\langle A, B, L, R, X \rangle$ satisfying the following.

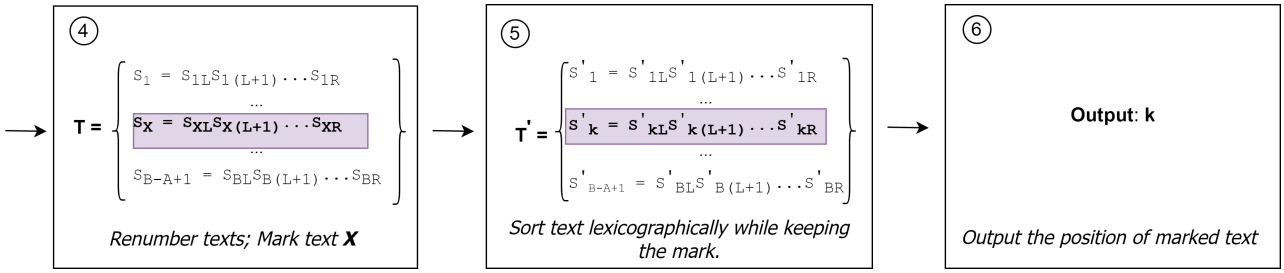
- $1 \leq A \leq B \leq N$
- $1 \leq L \leq R \leq M$
- $1 \leq X \leq B - A + 1$

For each task, Morgan should perform the following procedures.

1. Copy the original list of texts; let T be the copied list. This list will be updated throughout the task.
2. Remove all texts i from T that are **not** within the range of $A \leq i \leq B$.
3. For all remaining texts in T, remove all characters at index j that are **not** within the range of $L \leq j \leq R$;
4. The remaining texts in T is renumbered from 1 to $B - A + 1$. Mark text X in T.
5. Sort T lexicographically; let the result be T'. Note that the performed sort is a stable sort, meaning that if two texts are equal, then they maintain their order in the sorted list.
6. Output the position of the marked text in T'. The lexicographically smallest text will be at position 1 (one-based).

The image below are the ilustrations how the procedure works.





It turns out that it takes Morgan a lot of time to solve those tasks. Therefore, Adrian asks for your help to improve Morgan's program so that he can solve those tasks quickly and accurately.

A string s of length n is lexicographically smaller than string t with the same length if there exists an integer $1 \leq i \leq n$ such that $s_j = t_j$ for all $1 \leq j < i$, and $s_i < t_i$.

Input

Input begins with two integers N M ($1 \leq N, M \leq 100\,000$; $1 \leq N \times M \leq 100\,000$) representing the number of texts and the length of each text, respectively. Each of the next N lines contains a string S_i representing text i. Each text contains M lower-case characters.

The next line contains an integer Q ($1 \leq Q \leq 100\,000$) representing the number of tasks. Each of the next Q lines contains five integers A B L R X ($1 \leq A \leq B \leq N$; $1 \leq L \leq R \leq M$; $1 \leq X \leq B - A + 1$) representing a task.

Output

For each task, output an integer in a single line representing the answer of that task.

Sample Input #1

```
5 6
adrian
morgan
george
undine
stella
5
1 5 1 6 1
1 5 1 6 2
1 2 3 6 1
2 4 3 5 3
1 2 5 6 2
```

Sample Output #1

```
1
3
```

```
2
1
2
```

Explanation for the sample input/output #1

For tasks 1 and 2, the final T is all of the given texts. Task 1 marks text 1 of T, which is `adrian`; while task 2 marks text 2 of T, which is `morgan`. After sorted, the list T' becomes [`adrian`, `george`, `morgan`, `stella`, `undine`]. The marked text in task 1 is at position 1 in T'. Similarly, the marked text 2 is at posiition 3 in T'.

For task 3, the final T is [`rian`, `rgan`]. Task 3 marks text 1 of T, which is `rian`. After sorted, the list T' becomes [`rgan`, `rian`]. The marked text in task 3 is at position 2 in T'.

For task 4, the sorted list T' contains [`din`, `org`, `rga`]. Task 4 marks text 3 of T, `din`, which is at position 1 in T'.

For task 5, the sorted list T' contains [`an`, `an`]. Note that both `an` are different from each other; the first one is taken from text 1 of T, while the second one is taken from text 2 of T. Task 5 marks text 2 of T, `an` taken from text 2 of T, which is at position 2 of T'.

Sample Input #2

```
3 9
indonesia
nationaln
contestco
3
2 2 1 9 1
1 2 3 7 2
1 3 2 5 2
```

Sample Output #2

```
1
2
1
```

Problem I
Reporting Documents

Each citizen in ICPC Kingdom must have their N kingdom-issued documents, numbered from 1 to N, on their hands at any time. The guards often ask random citizens for their documents during their patrol.

As a citizen of ICPC Kingdom, Adrian also has these documents on his hands as well; however, some of them might be missing due to his negligence. The existence status of all of his documents are represented by a string B where B_i represents the existence of document i. If document i is on his hand, then $B_i = 1$. Otherwise, $B_i = 0$ if document i is missing.

For each of the next Q days, exactly one of the following scenarios will happen.

- 1 x. Adrian found his missing document x, so B_x is updated to 1 (it is guaranteed that $B_x = 0$ right before this scenario).
- 2 x. Adrian lost his document x, so B_x is updated to 0 (it is guaranteed that $B_x = 1$ right before this scenario).
- 3 x k. A guard asks Adrian for document $x + k \cdot i$, where $x \leq k$, for all i that satisfies $0 \leq i$ and $1 \leq x + k \cdot i \leq N$. For each document he couldn't provide when the guard asked for it, Adrian will be fined for 1 coin.

For each scenarios involving a guard (i.e. scenario 3), Adrian asks you to count how many coins he needs to pay for the fine.

Input

Input begins with an integer N ($1 \leq N \leq 200\,000$) representing the number of documents. The next line contains a string B of length N, where the i^{th} character of B is B_i ($B_i \in \{$ 0, 1$\}$), the initial existence status of document i.

The next line contains an integer Q ($1 \leq Q \leq 200\,000$) representing the number of days. Each of the next Q lines contains a scenario. Each scenario begins with an integer t ($t \in \{1, 2, 3\}$). If $t = 1$ or $t = 2$, then it is followed by an integer x ($1 \leq x \leq N$) representing scenario 1 or 2, respectively. It is guaranteed that integer x in scenarios 1 and 2 satisfy the scenario description. If $t = 3$, then it is followed by two integers x k ($1 \leq x \leq k \leq N$) representing scenario 3. There will be at least one scenario of type 3.

Output

For each scenario 3, output an integer in a single line representing how many coins Adrian needs to pay for the fine for that day.

Sample Input #1

```
10
1010001001
```

```
5
3 1 2
2 1
1 5
1 9
3 1 1
```

Sample Output #1

```
2
5
```

Explanation for the sample input/output #1

At first, Adrian only has documents 1, 3, 7, and 10 on his hand.

On day 1, a guard asks Adrian for documents $1 + 2 \cdot i$, i.e. documents 1, 3, 5, 7, and 9. Adrian doesn't have documents 5 and 9 on his hand, thus, he will be fined for 2 coins.

On day 2, 3, and 4, he lost document 1, found document 5, and found document 9, respectively.

On day 5, a guard asks Adrian for documents $1 + 1 \cdot i$, i.e. all documents from 1 to 10. Adrian doesn't have documents 1, 2, 4, 6, and 8 on his hand, thus, he will be fined for 5 coins.

Sample Input #2

```
25
00100000101001101000000101
10
3 2 4
1 5
1 21
2 11
2 5
3 1 5
1 5
3 5 5
3 3 8
3 1 25
```

Sample Output #2

```
5
4
2
2
1
```

Problem J
Finding Treasure

Morgan is designing a treasury for a treasure hunt event. The treasury can be represented as one dimensional grid with $N + 2$ cells, numbered from 0 to $N + 1$. Each cell can be empty or contain a treasure. At first, cell 0 and cell $N + 1$ contain a treasure, while the other cells are currently empty.

While experimenting with the treasury design, Morgan does Q updates, numbered from 1 to Q, to his treasury. In update i, he wants to change cell A_i. If cell A_i is empty, he will put a treasure on cell A_i. If cell A_i contains a treasure, he will remove the treasure from cell A_i and the cell becomes empty. It is guaranteed that $1 \leq A_i \leq N$ for all $1 \leq i \leq Q$, which implies cell 0 and cell $N + 1$ always contain a treasure.

After each change, Morgan wonders how difficult his treasure hunt is. He defines the *difficulty level* of standing on cell x as the multiplication of two values:

- the distance from x to the closest treasure on cell $y \leq x$, and
- the distance from x to the closest treasure on cell $y \geq x$.

The distance between two cells can be calculated as the absolute difference between the cell numbers. Then, the *total difficulty level* of his treasure hunt is defined as the sum of difficulty level of standing on cell x for all $0 \leq x \leq N + 1$.

Help Morgan to determine the total difficulty level of his treasure hunt after each update.

Input

Input begins with two integers N Q ($1 \leq N \leq 100\,000$; $1 \leq Q \leq 100\,000$) representing the size of the treasury room and the number of updates, respectively. Each of the next Q lines contains an integer A_i ($1 \leq A_i \leq N$) representing the cell that Morgan wants to change in update i.

Output

After each update that Morgan makes, output an integer in a single line representing the total difficulty level of his treasure hunts at that time.

Sample Input #1

```
3 3
1
3
1
```

Sample Output #1

```
4
1
```

```
4
```

Explanation for the sample input/output #1

For each cell that contains a treasure, the difficulty level of standing on that cell is 0.

After the first update, the difficulty level of standing on cell 2 and 3 are $|2-1|\times|2-4| = 2$ and $|3-1|\times|3-4| = 2$, respectively.

After the second update, the difficulty level of standing on cell 2 is $|2-1| \times |2-3| = 1$.

After the third update, the difficulty level of standing on cell 1 and 2 are $|1-3|\times|1-0| = 2$ and $|2-3|\times|2-0| = 2$, respectively.

Sample Input #2

```
10 14
7
2
9
5
2
9
6
7
5
8
6
8
7
10
```

Sample Output #2

```
66
31
23
8
23
31
30
40
55
40
88
220
66
60
```

Problem K
Permutation Construction

Adrian challenges you to construct a permutation of 1 to N (inclusive) that satisfies his requirements. Suppose that your permutation is denoted as P. There are N requirements, numbered from 1 to N. Requirement i is represented by A_i, which can be one of the followings:

- if $A_i = -1$, then there is no x that satisfy $x > i$ and $P_x > P_i$; or
- if $i < A_i \leq N$, then A_i is the smallest index larger than i that satisfy $P_{A_i} > P_i$.

Construct any permutation that satisfies all of the requirements, or tell Adrian if it is impossible to construct such a permutation.

Note that a permutation of 1 to N (inclusive) is a sequence where each integer from 1 to N appears in the sequence exactly once.

Input

Input begins with an integer N ($1 \leq N \leq 100\,000$). The next line contains N integers A_i ($A_i = -1$ or $i < A_i \leq N$) representing the given requirements.

Output

If at least one permutation that satisfies all the requirements can be found, output N space-separated integers in a single line representing the permutation. If there are several permutations that satisfy all of the requirements, output any of them.

If there is no permutation that satisfies all the requirements, output `-1` in a single line.

Sample Input #1

```
5
4 3 4 -1 -1
```

Sample Output #1

```
4 1 3 5 2
```

Sample Input #2

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

Sample Output #2

```
-1
```

This page is intentionally left blank.

Problem L
Expected Beauty

Morgan the robot has an array A of size N, indexed from 1 to N. The value of each element in A is randomly generated; A_i can be any integer from L_i to R_i (inclusive) with equal probability.

Morgan defines the *beauty* of A as follows. First, Morgan has a variable named `score` that is initialized to 0. An operation on the array a is as follows:

- Choose an index i such that $1 \leq i < |a|$ and $a_i = a_{i+1}$. If no such i exists, then the operation cannot be performed.
- Add the value of a_i to `score` and remove a_i from the array.
- The array a becomes the concatenation of the remaining elements without changing its order.

The *beauty* of A is the maximum value of $\texttt{score}^2$ Morgan can possibly get after performing zero or more operations on the array A.

Since the array is randomly generated, Morgan wonders about the expected beauty of A. Due to the inefficiency of his algorithm, Morgan asks for your help to calculate the expected value.

Input

Input begins with an integer N ($1 \leq N \leq 200\,000$) representing the size of array A. Each of the next N lines contains two integers L_i R_i ($1 \leq L_i \leq R_i \leq 10^8$).

Output

Let $M = 998\,244\,353$. It can be shown that the expected value can be expressed as an irreducible fraction $\frac{p}{q}$, where p and q are integers and $q \not\equiv 0 \mod M$. Output an integer x in a single line such that $0 \leq x < M$ and $x \cdot q \equiv p \mod M$.

Sample Input #1

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

Sample Output #1

```
831870298
```

Explanation for the sample input/output #1

There are 12 possibilities of A. Out of all possibilities, the following has positive beauty.

- $[1, 2, 2]$ with a beauty of 4.
- $[1, 3, 3]$ with a beauty of 9.
- $[2, 2, 1]$ with a beauty of 4.
- $[2, 2, 2]$ with a beauty of 16.
- $[2, 2, 3]$ with a beauty of 4.
- $[2, 3, 3]$ with a beauty of 9.

Therefore, the expected beauty of A is $(4 + 9 + 4 + 16 + 4 + 9)/12 = \frac{46}{12} = \frac{23}{6}$. Since $831\,870\,298 \cdot 6 \equiv 23 \mod 998\,244\,353$, you need to output $831\,870\,298$.

Sample Input #2

```
4
1 1
1 1
2 2
2 2
```

Sample Output #2

```
9
```

Explanation for the sample input/output #2

The only possible value of A is $[1, 1, 2, 2]$ with a beauty of $(1 + 2)^2 = 9$.

Sample Input #3

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

Sample Output #3

```
0
```

Problem M
Obstacle Course

As a businessman and parkour enthusiast, Morgan owns an obstacle course consisting of N road segments numbered from 1 to N. These road segments are arranged sequentially such that road 1 is adjacent to road 2, road 2 is adjacent to road 3, ..., road $N-1$ is adjacent to road N. Currently, each odd-numbered road segment has a height of a positive integer A_i, and each even-numbered road segment has a height of 0. In Morgan's course, anyone can start from any road segment, parkour towards one direction, and stop at any road segment as long as they can reach that road segment.

As a beginner athlete, Adrian thinks that Morgan's obstacle course may be too hard due to the height difference between any two consecutive roads. Adrian enjoys doing parkour whenever the height difference between two consecutive roads is exactly 1. If the next road segment has a difference of more than 1 from the current road segment, then Adrian cannot climb or jump down to the next road segment and will stop his parkour activity. On the other hand, if the next road segment has the same height as the current road segment, then Adrian will lose interest and stop his parkour activity. He will also stop his parkour activity if he reaches the end of the obstacle course, i.e. road segment N or road segment 1.

As his friend, Morgan wants Adrian to be happy with the course. Adrian's happiness is defined as the number of road segments that he parkoured before he lost interest and stops his parkour activity. Adrian will start from any road segment that will give him the largest happiness. Based on these facts, Morgan decided to change the heights of some (possibly none) **even-numbered** road segments such that Adrian's largest happiness for the course is maximized. However, before he makes any changes, he asked you to determine the largest possible Adrian's happiness that can be obtained after changing the height.

For example, let $A_{1..9} = [6, 0, 4, 0, 8, 0, 12, 0, 14]$. By changing A into $[6, 5, 4, 5, 8, 11, 12, 13, 14]$, Adrian can do parkour from segment 1 to 4 or from segment 6 to 9, and his happiness will be 4, which is also the largest happiness after that change. There are some other changes that can make Adrian's largest happiness equal to 4 as well, e.g. $[6, 5, 4, 3, 8, 7, 12, 12, 14]$, but no change can make Adrian's largest happiness larger than 4.

Help Morgan to determine the largest possible Adrian's happiness in his new course.

Input

Input begins with an integer N ($2 \leq N \leq 100\,000$) representing the number of road segments. The next line contains N integers A_i ($A_i = 0$ if i is even; $1 \leq A_i \leq 100\,000$ if i is odd) representing the height of segment i from 1 to N respectively.

Output

Output an integer in a single line that represents the largest possible Adrian's happiness in Morgan's new course.

Sample Input #1

```
9
6 0 4 0 8 0 12 0 14
```

Sample Output #1

```
4
```

Explanation for the sample input/output #1

This is the example from the problem description.

Sample Input #2

```
7
3 0 5 0 5 0 7
```

Sample Output #2

```
7
```

Explanation for the sample input/output #2

By changing A to $[3, 4, 5, 6, 5, 6, 7]$ Adrian's happiness will be 7.

Sample Input #3

```
6
6 0 2 0 8 0
```

Sample Output #3

```
3
```

Sample Input #4

```
7
4 0 6 0 10 0 9
```

Sample Output #4

```
4
```