

Problem Set

Please check that you have 11 problems that are spanned across 27 pages in total (including Korean translation and this cover page).

A. Citation Index	(2+2 pages)	Korean translation available
B. Coloring	(2 pages)	
C. Convex Hull	(2 pages)	
D. Cross Roller	(2 pages)	
E. Eating Candies	(1 pages)	
F. Interactive String Matching	(3 pages)	
G. Korean Characters	(2+2 pages)	Korean translation available
H. Marathon	(2 pages)	
I. Pumpkin Friends	(2 pages)	
J. Secret Agent	(2 pages)	
K. Square	(1+1 pages)	Korean translation available



Problem A

Citation Index

Time Limit: 1 Second

A digital library service maintains a citation database for the academic community and reports a single number for each registered researcher that summarizes the impact of the researcher's work.

The number reported by the service is the *h-index*, a widely used measure of research impact. Consider a researcher who has written n papers, and let c_i be the number of times the i -th paper has been cited. The *h-index* of the researcher is the largest integer h such that at least h of the n papers have been cited h or more times. Note that $h = 0$ always satisfies this condition, so the *h-index* is well defined for every researcher.

For example, suppose that a researcher has written five papers whose citation counts are 5, 4, 3, 2, and 1. Three of the papers have been cited 3 or more times, so $h = 3$ satisfies the condition. On the other hand, only two of them have been cited 4 or more times, so $h = 4$ does not. The *h-index* of this researcher is therefore 3.

A citation database is never static. Newly published articles keep citing the papers already in the database, and the service updates the citation counts as it collects them. Each update specifies a paper and the number of new citations to add to it. Updates are cumulative: each update is applied to the citation counts resulting from all previous updates. Since the reported number must always reflect the current state of the database, the service recomputes the *h-index* right after applying each update.

You are given the initial citation counts of a researcher's n papers, followed by q updates in chronological order.

Write a program that reports the *h-index* of the researcher immediately after each of the q updates is applied.

Input

Your program is to read from standard input. The input starts with a line containing two integers n and q ($1 \leq n, q \leq 2 \times 10^5$), where n is the number of papers written by the researcher and q is the number of updates.

The second line contains n integers $c_1, c_2, \dots, c_n$ ($0 \leq c_i \leq 10^9$), where c_i is the initial number of citations of the i -th paper.

Each of the following q lines contains two integers i and x ($1 \leq i \leq n$; $1 \leq x \leq 10^9$), meaning that the number of citations of the i -th paper increases by x . The updates are given in the order in which they are applied. Note that the number of citations of a paper may not fit in a 32-bit integer.

Output

Your program is to write to standard output. Print q lines. The j -th line should contain the *h-index* of the researcher immediately after the j -th update is applied. The *h-index* of the initial state is not printed.

The following shows the sample input and output for two test cases.

Sample Input 1

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

Output for the Sample Input 1

3
3
4

Sample Input 2

3 1
1000000000 1 0
2 999999999

Output for the Sample Input 2

2

Problem A

Citation Index

제한 시간: 1 초

어떤 디지털 도서관 서비스가 학술 공동체를 위해 인용 데이터베이스를 관리한다. 이 서비스는 등록된 연구자마다 연구 영향력을 요약하는 값 하나를 보여 준다.

서비스가 보여 주는 값은 h -index이며, 연구의 영향력을 나타내는 데 널리 쓰이는 지표이다. 논문을 n 편 쓴 연구자를 생각하자. i 번째 논문이 인용된 횟수를 c_i 라고 하면, 이 연구자의 h -index는 인용 횟수가 h 이상인 논문이 h 편 이상이라는 조건을 만족하는 가장 큰 정수 h 이다. $h = 0$ 은 이 조건을 항상 만족하므로 h -index는 언제나 정의된다.

예를 들어 어떤 연구자가 논문 다섯 편을 썼고 인용 횟수가 각각 5, 4, 3, 2, 1이라고 하자. 이 가운데 3번 이상 인용된 논문이 세 편이므로 $h = 3$ 은 조건을 만족한다. 반면 4번 이상 인용된 논문은 두 편뿐이므로 $h = 4$ 는 조건을 만족하지 않는다. 따라서 이 연구자의 h -index는 3이다.

인용 데이터베이스는 한 번 만들어 두고 끝나지 않는다. 새로 나온 논문이 데이터베이스에 있는 논문을 계속 인용하고 서비스는 이를 확인해 인용 횟수를 갱신한다. 각 갱신은 논문 한 편과 그 논문에 추가할 인용 횟수를 지정한다. 갱신 결과는 누적되며, 각 갱신은 앞선 모든 갱신이 반영된 인용 횟수에 적용된다. 서비스가 보여 주는 값은 데이터베이스의 현재 상태를 반영해야 하므로, 갱신을 하나 적용할 때마다 곧바로 h -index를 다시 계산한다.

어떤 연구자가 쓴 n 편의 논문에 대한 초기 인용 횟수와 q 개의 갱신이 시간 순서대로 주어진다.

갱신을 하나 적용할 때마다 그 직후의 h -index를 출력하는 프로그램을 작성하시오.

Input

입력은 표준 입력을 사용한다. 첫 줄에 연구자가 쓴 논문의 수 n 과 갱신의 수 q ($1 \leq n, q \leq 2 \times 10^5$)가 주어진다.

두 번째 줄에 정수 n 개 $c_1, c_2, \dots, c_n$ ($0 \leq c_i \leq 10^9$)이 주어진다. c_i 는 i 번째 논문의 초기 인용 횟수이다.

이어지는 q 개의 줄에는 각각 두 정수 i 와 x ($1 \leq i \leq n$; $1 \leq x \leq 10^9$)가 주어진다. i 번째 논문의 인용 횟수가 x 만큼 늘어난다는 뜻이다. 갱신은 적용되는 순서대로 주어진다. 논문 한 편의 인용 횟수는 32비트 정수형의 범위를 벗어날 수 있다.

Output

출력은 표준 출력을 사용한다. q 개의 줄을 출력한다. j 번째 줄에는 j 번째 갱신을 적용한 직후의 h-index를 출력한다. 초기 상태의 h-index는 출력하지 않는다.

다음은 두 테스트 케이스에 대한 입출력 예이다.

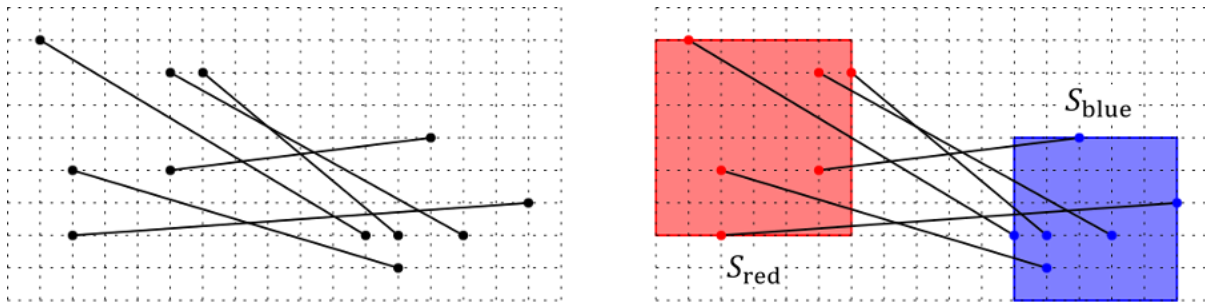
Sample Input 1	Output for the Sample Input 1
5 3 1 3 2 5 4 1 2 3 5 2 1	3 3 4
Sample Input 2	Output for the Sample Input 2
3 1 1000000000 1 0 2 999999999	2

Problem B

Coloring

Time Limit: 1 Second

You are given n line segments in the plane. For each segment, color one endpoint red and the other blue. After coloring all endpoints in this way, let S_{red} and S_{blue} be smallest squares enclosing all red and all blue points, respectively. All squares in this problem are axis-aligned; that is, each edge is parallel to either the x -axis or the y -axis. Note also that any point in the plane is regarded as a square of side length zero.



The above figure shows an example with $n = 6$ line segments (left) and a possible coloring of their endpoints together with two squares S_{red} (in red) and S_{blue} (in blue) enclosing all red and all blue points, respectively (right).

Write a program that finds a coloring minimizing the maximum of the side lengths of S_{red} and S_{blue} .

Input

Your program is to read from standard input. The input starts with a line containing an integer n ($1 \leq n \leq 500,000$), where n is the number of input line segments $s_1, s_2, \dots, s_n$.

The next n lines describe $s_1, s_2, \dots, s_n$ in order. For $1 \leq i \leq n$, the i -th of these lines contains four integers x_i, y_i, x'_i, y'_i ($-1,000,000 \leq x_i, y_i, x'_i, y'_i \leq 1,000,000$), where (x_i, y_i) and (x'_i, y'_i) are the two endpoints of s_i .

Note that the two endpoints of a segment may coincide; that is, $x_i = x'_i$ and $y_i = y'_i$ is allowed. Note also that the endpoints of different segments may coincide. See the sample input and output below.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the minimum possible value of the maximum of the side lengths of S_{red} and S_{blue} , over all colorings described above.

The following shows the sample input and output for three test cases.

Sample Input 1

6
1 8 11 2
2 2 16 3
2 4 12 1
5 4 13 5
5 7 14 2
6 7 13 2

Output for the Sample Input 1

6

Sample Input 2

4
1 1 2 2
2 2 1 1
1 1 1 1
2 2 2 2

Output for the Sample Input 2

1

Sample Input 3

4
1 1 2 2
2 2 1 1
1 1 2 2
2 2 1 1

Output for the Sample Input 3

0

Problem C

Convex Hull

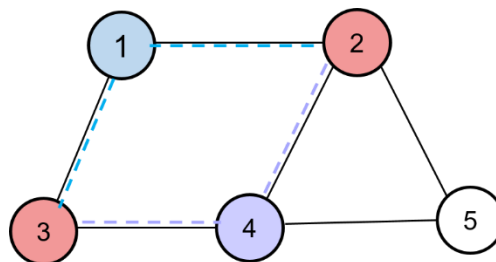
Time Limit: 2 Seconds

The convex hull is a well-known concept in computational geometry for points in the plane. The same concept can be extended to graphs.

Let $G = (V, E)$ be an undirected simple graph with a set V of n vertices and a set E of m edges. The graph G is connected, that is, there is a path between any two vertices in G . The length of a path is the number of edges it contains. For two vertices u and v , let $I(u, v)$ be the set of all vertices w such that w lies on a shortest path between u and v . Then, for a subset $S \subseteq V$, define $I(S) = \bigcup_{u, v \in S} I(u, v)$, which is a set of vertices containing all vertices of S and every vertex outside S that lies on a shortest path between any two vertices of S .

A subset $S \subseteq V$ is called *convex* if $I(S) = S$. The convex hull $\mathcal{H}(S)$ of S is the smallest convex set containing S . Thus, if S is convex, $\mathcal{H}(S) = S$. It is known that $\mathcal{H}(S)$ exists and is unique for every $S \subseteq V$.

For example, the figure below shows a graph with 5 vertices. Let $S = \{2, 3\}$. Then $I(S) = \{1, 2, 3, 4\}$, because $2 - 1 - 3$ and $2 - 4 - 3$ are the two shortest paths between 2 and 3. Thus, S is not convex. On the other hand, $\{1, 2, 3, 4\}$ is convex and is the smallest convex set containing S . Hence, $\mathcal{H}(S) = \{1, 2, 3, 4\}$.



Given a graph G with n vertices and m edges and a set S of vertices in G , write a program to output the size of the convex hull $\mathcal{H}(S)$ of S and all the vertices in $\mathcal{H}(S)$.

Input

Your program is to read from standard input. The input starts with a line containing three integers, n , m and k ($2 \leq n \leq 500$; $1 \leq m \leq n(n-1)/2$; $1 \leq k \leq n$), where n is the number of vertices, m is the number of edges in a connected graph G , and k is the number of vertices in the set S . The vertices are numbered from 1 to n .

The second line contains k distinct integers between 1 and n , inclusive, which represent the vertices belonging to S .

Each of the following m lines contains two distinct integers between 1 and n , inclusive, representing an edge of G . It is guaranteed that no two edges connect the same pair of vertices.

Output

Your program is to write to standard output. Print exactly two lines. The first line should contain the size of the convex hull $\mathcal{H}(S)$ of S . The second line contains the vertices belonging to $\mathcal{H}(S)$ in increasing order.

The following shows the sample input and output for three test cases.

Sample Input 1

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

Output for the Sample Input 1

Sample Input 2

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

Output for the Sample Input 2

Sample Input 3

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

Output for the Sample Input 3

Problem D

Cross Roller

Time Limit: 0.5 Seconds

Consider a game in which you paint an $n \times n$ grid using a special tool called a “Cross Roller.” In each operation, you choose a cell (r, c) and a color, and the roller simultaneously paints the r -th row and the c -th column with that color. If a cell has already been painted, the new color simply overwrites the previous color. All cells are initially unpainted and appear white. Once a cell is painted, it can never revert to being unpainted. Note that the tool can be used any number of times (including zero), and the same cell can be chosen repeatedly.

For example, painting the cell $(1, 1)$ yellow, followed by the cell $(2, 2)$ blue, produces the valid pattern shown in the figure on the left (Sample Input 1). In contrast, the pattern on the right (Sample Input 2) is impossible to create using the Cross Roller.



Write a program to determine whether a given target pattern on the grid can be created using this tool.

Input

Your program is to read from standard input. The input starts with a line containing two integers n and k ($1 \leq n \leq 1,200$; $1 \leq k \leq 50$), representing the size of the grid and the number of available colors, respectively. Each color is represented by an integer from 1 to k , inclusive, while 0 represents an unpainted cell. Each of the following n lines contains n integers between 0 and k , representing the target pattern row by row.

Output

Your program is to write to standard output. If the target pattern cannot be created using the Cross Roller, print exactly one line containing NO. If the target pattern can be created, print YES on the first line. On the second line, print an integer m ($0 \leq m \leq 10^5$), representing the number of painting operations. The following m lines should each contain three space-separated integers r , c , and x ($1 \leq r, c \leq n$; $1 \leq x \leq k$), indicating an operation that selects the cell (r, c) and the color x . Starting from an entirely unpainted grid, applying these m operations in the given order must result in exactly the target pattern. If several valid sequences of operations exist, printing any one of them will be accepted.

It can be shown that whenever the target pattern can be created, there exists a sequence of operations satisfying all the above output requirements.

The following shows the sample input and output for three test cases.

Sample Input 1

3 2
1 2 1
2 2 2
1 2 0

Output for the Sample Input 1

YES
2
1 1 1
2 2 2

Sample Input 2

3 3
1 3 1
3 1 3
1 3 1

Output for the Sample Input 2

NO

Sample Input 3

5 50
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0

Output for the Sample Input 3

YES
0

Problem E

Eating Candies

Time Limit: 1 Second

Bay and Rei have n candy skewers. On each skewer, blue candies (B) and red candies (R) are arranged in a row. Bay eats only blue candies, and Rei eats only red candies. To share the candies, they play the following game.

Bay and Rei alternately take turns. On each turn, the player chooses one skewer, removes a candy from one of the two ends of the skewer, and eats it. Note that Bay can remove only blue candies, and Rei can remove only red candies. If a player cannot eat any candy on their turn, that player loses the game.

For example, suppose that there is only one skewer BR and Bay moves first. Bay eats the blue candy at the left end, and then Rei eats the remaining red candy. Now Bay cannot eat any candy, so Rei wins the game. See Sample Input/Output 1 below.

Given the n skewers and the player who moves first, write a program to determine the winner, assuming that both players play optimally.

Input

Your program is to read from standard input. The input starts with a line containing an integer n ($1 \leq n \leq 1,000$) and a character c , where n is the number of skewers and c is B if Bay moves first or R if Rei moves first. Each of the following n lines contains a string consisting only of the characters B and R, which represents the candies on a skewer in order from its left end to its right end. The length of each string is at least 1, and the sum of the lengths of all n strings is at most 5,000.

Output

Your program is to write to standard output. Print exactly one line. The line should contain B if Bay wins, or R if Rei wins.

The following shows the sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
1 B BR	R
Sample Input 2	Output for the Sample Input 2
2 B BBR RBBBR	B
Sample Input 3	Output for the Sample Input 3
1 R RRBBR	R

Problem F

Interactive String Matching

Time Limit: 5 Seconds

You are given two strings S and T . Let $|S|$ be the length of S and $|T|$ the length of T . Given that $|S| \geq |T|$, find every index $1 \leq a \leq |S| - |T| + 1$ such that the length $|T|$ substring of S starting at index a is equal to T .

Interaction

This is an interactive problem. Your submitted program will interact with an interactor inside the grading server, which reads input from and writes output to your program. Your program must gather information by printing queries and reading the interactor's replies. After every line of output, your program must immediately flush the output.

Initial Input

At the beginning of the interaction, the interactor provides two positive integers $|S|$ and $|T|$ respectively, separated by a space ($1 \leq |T| \leq |S| \leq 200,000$).

Queries

Your program may repeatedly output exactly one of the following commands until the interactor outputs "correct" or "wrong". Every number and special character ("?", "!") used in the query should be separated by a single space.

- "? i j "
 - This query compares the i -th character of S with the j -th character of T . The indices of both strings start at 1.
 - If at least one of the characters does not exist, the interactor outputs "wrong".
- "! k a_1 a_2 $\dots$ a_k "
 - Report $a_1, a_2, \dots, a_k$, which must be exactly all indices a such that the length- $|T|$ substring of S starting at a is equal to T .
 - Your program can print "! 0" if it concludes that no match is found.
 - Both $0 \leq k \leq |S| - |T| + 1$ and $1 \leq a_1 < a_2 < \dots < a_k \leq |S| - |T| + 1$ must hold. If not, the interactor outputs "wrong" **even if your program printed all indices that satisfy the condition.**

Each command must be printed on its own line and immediately flushed. The total number of the queries of the **first** kind issued **must not exceed** $2 \times (|S| + |T|)$. Your program must issue the query of the second kind **exactly once**.

Interactor Responses

After each command, the interactor responds as follows:

- For the queries of the first kind, if the two characters are the same, the interactor outputs “1”. If not, the interactor outputs “0”.
- For the queries of the second kind, if your program reported exactly all matching indices $a_1 < a_2 < \dots < a_k$, the interactor outputs “correct”. The interactor outputs “wrong” in all other cases.
- The interactor immediately terminates the interaction after printing “wrong” if any rule is violated — for example, a malformed command, an out-of-range index, or exceeding $2 \times (|S| + |T|)$ queries of the first kind issued.

Receiving “correct” on a query starting with “!” means that your program has successfully found all indices. Receiving “wrong” at any point means that your program has failed. In either case, your program should terminate gracefully. **Do not forget to flush the output** after printing each command.

The interactor **may be adaptive**. The interactor only guarantees that there exists at least one pair of strings (S, T) consistent with all queries and responses so far. You must **not** assume that the characters of S and T belong to the Latin alphabet.

The time and memory used by the interactor are also included in the calculation of your program’s execution time and memory usage. You can assume that the maximum time used by the interactor is 1.5 seconds and the maximum amount of memory is 64 MiB.

The following shows a sample interaction when S is “abacabacaba” and T is “abaca”.

Read (Interactor’s Response)	Sample Interaction 1	Write (Your Command)
1 1 5		
	? 1 3	
1		
	? 2 2	
1		
	? 5 4	
0		
	? 7 3	
1		
	? 10 5	
0		
	? 6 3	
0		
	? 11 1	
1		
	? 9 4	
0		
	! 2 1 5	
correct		

To flush, your program needs to do the following right after writing a command and a newline:

- `fflush(stdout)` in C;
- `std::cout << std::flush` in C++;
- `System.out.flush()` in Java or Kotlin;
- `sys.stdout.flush()` in Python.

A testing tool is provided to help you develop your solution, so it is not mandatory to use it. If you want to use it, you can download the attachment `testing_tool.py` from the *DOMjudge Problemset* page. You can run the testing tool to see how your program interacts with a specific set of strings. **This testing tool can be used regardless of the language of your program as follows.** The same explanation is also included in the comments of `testing_tool.py`.

Usage: `python3 testing_tool.py -f <inputfile> -- <program>`

Use the `-f` parameter to specify the input file, e.g. `input.in`.

Format of the input file: The file should consist of exactly two lines. The first line contains S and the second line T .

Example: The strings from the sample interaction are represented as follows.

abacabacaba abaca

The tool is provided as-is, and you should feel free to make whatever alterations or augmentations you like to it. **Note that it is not guaranteed that a program that passes the testing tool will be accepted.** For example, the testing tool **does not verify** whether your answer is exhaustive; missing indices are not reported.

If you have a C++ solution stored in a file called “`sol.cpp`”, you must first compile using “`g++ sol.cpp -o sol`” and then invoke the testing tool with:

```
python3 testing_tool.py -f input.in -- ./sol
```

If you have a Python solution that you would run using “`pypy3 solution.py`”, you can invoke the testing tool with:

```
python3 testing_tool.py -f input.in -- pypy3 solution.py
```

If you have a Java solution that you would run using “`java MyClass`”, you can invoke the testing tool with:

```
python3 testing_tool.py -f input.in -- java MyClass
```

If you have a Kotlin solution that you would run using “`kotlin SolutionKt`”, you can invoke the testing tool with:

```
python3 testing_tool.py -f input.in -- kotlin SolutionKt
```

Problem G

Korean Characters

Time Limit: 1 Second

Alan is having trouble with the file names in Korean on his MacBook. When copying the files to Windows, the Korean characters in the file names are decomposed into graphemes. For example, the file name “문서.txt” on MacBook is represented as “ㄹㅁㅊㄴㅅ ㅈ .txt” in Windows.

Write a program to help Alan. Specifically, your task is to form a valid Korean file name from a sequence of decomposed Korean letters. To simplify the problem, assume the Korean characters of the file names do not include compound vowels (e.g. ㅏ, ㅑ), tense consonants (e.g. ㅊ, ㅌ), or compound final consonants (e.g. ㅃ, ㅆ). For example, characters such as “위” or “닭” are excluded in the file names.

Each Korean grapheme is represented by a Latin letter as follows:

ㄱ: g	ㄴ: n	ㄷ: d	ㄹ: r	ㅁ: m	ㅂ: b	ㅅ: s
ㅇ: x	ㅈ: j	ㅊ: c	ㅋ: k	ㅌ: t	ㅍ: p	ㅎ: h
ㅏ: a	ㅑ: A	ㅓ: e	ㅕ: E	ㅗ: o	ㅛ: O	ㅜ: u
ㅟ: U	ㅡ: w	ㅣ: i				

In this problem, a Korean character is valid if it consists of a consonant followed by a vowel, optionally followed by another consonant.

The formed Korean characters are separated by hyphens. For example, the file name “ㄹㅁㅊㄴㅅ ㅈ” is represented as “munse”. After forming the Korean characters, it becomes “mun-se”, which represents “문서”.

Input

Your program is to read from standard input. The input consists of a single line containing a string of uppercase and lowercase Latin letters. The string is expected to be the Latin-letter representation of a decomposed Korean file name; your program should report an error otherwise. The string length n is less than 80 ($1 \leq n < 80$).

Output

Your program is to write to standard output. If the input string is a valid decomposed Korean file name, print the Latin-letter representation of the formed Korean file name; the Korean characters in the output should be separated by hyphens. If the input is not a valid Korean name, print “-ERROR-” instead.

The following shows the sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
munse	mun-se
Sample Input 2	Output for the Sample Input 2
hangwrxirwm	han-gwr-xi-rwm
Sample Input 3	Output for the Sample Input 3
English	-ERROR-

Problem G

Korean Characters

제한 시간: 1 초

앨런(Alan)은 맥북(MacBook)의 한글 파일 이름 때문에 어려움을 겪고 있다. 한글로 된 이름의 파일을 윈도우즈(Windows)로 복사하면 한글 글자가 자소가 분리된 형태로 나타나기 때문이다. 예컨대 맥북의 파일 이름이 "문서.txt"라면 윈도우즈에서는 "문서.txt"로 표기된다.

앨런을 도와줄 프로그램을 작성해 보자. 구체적으로, 자소가 분리된 한글 이름을 올바른 한글 파일 이름으로 나타내고자 한다. 문제를 간단히 하기 위해 이중 모음(ㅑ, ㅕ 등)이나 쌍자음(ㅃ, ㅆ 등), 겹받침(예: ㅁ, ㄹ)은 나타나지 않는다고 하자. 예컨대, "위"나 "닭"과 같은 글자는 파일 이름에 나타나지 않는다.

한글 자소를 영어 알파벳으로 나타내기 위해 다음과 같은 규칙으로 적는다.

ㄱ: g	ㄴ: n	ㄷ: d	ㄹ: r	ㅁ: m	ㅂ: b	ㅅ: s
ㅇ: x	ㅈ: j	ㅊ: c	ㅋ: k	ㅌ: t	ㅍ: p	ㅎ: h
ㅏ: a	ㅑ: A	ㅓ: e	ㅕ: E	ㅗ: o	ㅛ: O	ㅜ: u
ㅠ: U	ㅡ: w	ㅣ: i				

이 문제에서 한글 한 글자는 초성 자음 하나, 중성 모음 하나, 그리고 선택적으로 종성 자음 하나로 이루어진다.

모아쓰기로 모은 한글 글자는 하이픈(-)으로 구분하여 나타낸다. 예를 들어, "문서.txt"는 "munse"로 나타내며, 이를 모아쓰면 "mun-se"가 된다. 여기서 "mun-se"는 "문서"를 나타내는 영문자 표현이다.

Input

입력은 표준 입력을 사용한다. 입력은 한 행으로 구성되며, 영어 알파벳 대소문자만으로 이루어진 문자열이 주어진다. 주어진 영어 문자열은 자소가 분리된 한글 이름의 영문자 표현이어야 하나, 아니라면 오류를 보고해야 한다. 입력의 문자열 길이 n 은 80 미만이다($1 \leq n < 80$).

Output

출력은 표준 출력을 사용한다. 입력 문자열이 자소를 모아쓴 한글 파일 이름 표현이라면, 한글 이름의 영문자 표현을 출력한다. 한글 이름의 영문자 표현에서는 한글의 글자 사이를 하이픈으로 구별한다. 입력의 영문자 표현이 적절한 한글 이름이 아니라면 "-ERROR-"를 출력한다.

다음은 세 테스트 케이스에 대한 입출력 예이다.

Sample Input 1	Output for the Sample Input 1
munse	mun-se
Sample Input 2	Output for the Sample Input 2
hangwrxirwm	han-gwr-xi-rwm
Sample Input 3	Output for the Sample Input 3
English	-ERROR-

Problem H

Marathon

Time Limit: 1 Second

The International Marathon Association (IMA) is planning a marathon course that passes through the downtown area of a famous tourist destination. Along the straight road, a beautiful coastline stretches out on one side, while historic buildings line the other.

Some roadside buildings present potential psychological hazards to marathon runners—such as the risk of falling debris due to ongoing construction or excessive noise. Consequently, the IMA assesses the condition of each building to determine the potential risk level posed to runners and assigns a risk rating to each. These risk ratings are represented by non-negative integers.

The IMA has decided to deploy security personnel to mitigate the risk associated with buildings that have high risk ratings. If t security personnel are stationed in a building with a risk level of d , the risk level becomes $\max(0, (d - t))$.

There are n buildings in total along the road. The IMA plans to select a section consisting of l ($l \leq n$) consecutive buildings as the marathon section. The buildings are numbered from 1 to n , and the risk level of Building i is denoted by d_i , a non-negative integer. For a consecutive building section $[i, j]$ ($i \leq j$), its *section risk* is defined as $\max(d_i, d_{i+1}, \dots, d_{j-1}, d_j)$.

For example, suppose $n = 6$ and $l = 4$, and the initial risk levels of the 6 buildings are $(7, 2, 3, 5, 1, 6)$, respectively. There are three sections of length 4, and the section risk of each section is as follows. Note that the length of section $[i, j]$ is $(j - i + 1)$, that is the number of buildings in it.

Section	[1, 4]	[2, 5]	[3, 6]
Section risk	7	5	6

Assume that four security personnel are deployed, one assigned to Building 3 and three to Building 4. Then, the risk levels of the six buildings change to $(7, 2, \mathbf{2}, \mathbf{2}, 1, 6)$, and the section risks change as follows.

Section	[1, 4]	[2, 5]	[3, 6]
Section risk	7	2	6

The IMA intends to deploy k security personnel among the n buildings and select a section of length l as the marathon section so as to minimize its section risk.

Write a program to compute **the minimum possible** section risk, over all ways of deploying the security personnel and all choices of the section of length l .

Input

Your program is to read from standard input. The input starts with a line containing three integers n , l , and k ($1 \leq l \leq n \leq 2 \times 10^5$; $0 \leq k \leq 10^9$), where n is the total number of buildings, l is the length of the marathon section, and k is the number of security personnel to be deployed.

The next line contains n integers $d_1, d_2, \dots, d_n$ ($0 \leq d_i \leq 10^9$), where d_i is the initial risk level of Building i .

Output

Your program is to write to standard output. Print exactly one line. The line should contain the minimum possible section risk.

The following shows the sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
5 2 1 1 2 3 4 5	1
Sample Input 2	Output for the Sample Input 2
5 4 3 3 4 5 4 3	4
Sample Input 3	Output for the Sample Input 3
7 4 5 10 11 10 11 10 10 10	9

Problem I

Pumpkin Friends

Time Limit: 1 Second

Spooky, a little ghost who loves pumpkins, has n pumpkin friends planted in a single row. The pumpkins are numbered from 1 to n from left to right. For each pumpkin i , Spooky assigns a positive integer a_i , called its **joy value**.

For an interval $[l, r]$ of the pumpkin field, where $1 \leq l \leq r \leq n$, its total joy is

$$\sum_{i=l}^r a_i.$$



Initially, no pumpkin has been smashed. One day, the fairy queen *Erin* gets angry with Spooky and decides to smash all the pumpkins, one by one. The order in which she smashes them is given by a permutation $p_1, p_2, \dots, p_n$. At step t , she smashes pumpkin p_t . Once a pumpkin is smashed, it remains smashed.

At any moment, Spooky defines the **joy score** of her pumpkin field as the number of intervals $[l, r]$ satisfying both of the following conditions:

- None of the pumpkins $l, l + 1, \dots, r$ have been smashed, and
- The total joy of the interval is between L and R , inclusive.

Given the joy values of the pumpkins and the order in which Erin smashes them, write a program to compute the joy score of the pumpkin field before any pumpkin is smashed and after each of the n steps.

Input

Your program is to read from standard input.

The first line contains three integers n , L , and R ($1 \leq n \leq 500,000$; $1 \leq L \leq R \leq 10^{15}$), where n is the number of pumpkins, and L and R are the lower and upper bounds, respectively, on the total joy of an interval counted in the joy score.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq 10^9$), where a_i is the joy value of pumpkin i .

The third line contains n integers $p_1, p_2, \dots, p_n$, which form a permutation of $1, 2, \dots, n$. Pumpkin p_t is smashed at step t .

Output

Your program is to write to standard output.

Print exactly one line containing $n + 1$ integers $c_0, c_1, \dots, c_n$ separated by spaces.

For each t with $0 \leq t \leq n$, c_t is the joy score of the pumpkin field after pumpkins $p_1, p_2, \dots, p_t$ have been smashed. In particular, c_0 is the joy score before any pumpkin is smashed.

The following shows the sample input and output for three test cases.

Sample Input 1

7 4 8
2 5 4 4 4 5 5
6 1 5 2 3 4 7

Output for the Sample Input 1

9 8 7 5 4 2 1 0

Sample Input 2

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

Output for the Sample Input 2

8 4 2 1 1 1 0

Sample Input 3

1 2 3
5
1

Output for the Sample Input 3

0 0



Problem J

Secret Agent

Time Limit: 1.5 Seconds

You are the Director of the International Counterintelligence and Protection Command (ICPC). You are planning an operation to exfiltrate classified documents from a hostile area by deploying n secret agents into the operational zone.

The i -th agent stays in the operational zone during the time interval $[s_i, e_i]$ and has a capability rating of w_i . Two agents i and j can hand over documents to each other if their stay intervals overlap for at least an instant, that is, if $\max(s_i, s_j) \leq \min(e_i, e_j)$.

For organizational efficiency, **no agent's stay interval completely contains another agent's interval.**

To extract the documents, you need to select a sequence of k distinct agents $v_1, v_2, \dots, v_k$ ($k \geq 1$) to form a relay line. The documents will start with agent v_1 , be passed sequentially through $v_2, v_3, \dots, v_{k-1}$, and finally be taken out of the zone by agent v_k . The relay line must satisfy the following two conditions.

- **Feasibility (Connectivity):** Adjacent agents in the line, v_i and v_{i+1} , must have overlapping stay intervals for all $1 \leq i \leq k - 1$.
- **Confidentiality (Security):** Non-adjacent agents in the line, v_i and v_j with $j - i \geq 2$, must not have overlapping stay intervals. This is to ensure that if any agent is captured, the entire relay line is not exposed.

Agents not included in the relay line do not participate in the operation, so their stay intervals can overlap with other agents freely.

You want to maximize the sum of the capability ratings of the agents in the relay line, $\sum_{i=1}^k w_{v_i}$. Write a program to find the maximum possible sum of capabilities among all valid relay lines.

Input

Your program is to read from standard input. The first line contains an integer n ($2 \leq n \leq 8,000$), where n is the number of secret agents.

For each $1 \leq i \leq n$, the i -th of the following n lines contains three integers s_i, e_i , and w_i ($1 \leq s_i < e_i \leq 2n$; $1 \leq w_i \leq 10^5$), representing the stay interval $[s_i, e_i]$ and the capability rating w_i of the i -th agent. All $2n$ integers $s_1, s_2, \dots, s_n, e_1, e_2, \dots, e_n$ are **distinct**. For all $i \neq j$, if $s_i < s_j$, then $e_i < e_j$, which means no agent's interval completely contains another agent's interval.

Output

Your program is to write to standard output. Print exactly one line. The line should contain a single integer representing the maximum possible sum of capability ratings of the agents in a valid relay line.

The following shows the sample input and output for four test cases.

Sample Input 1

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

Output for the Sample Input 1

```
13
```

Sample Input 2

```
6
2 6 7
9 12 9
3 8 20
5 10 5
7 11 6
1 4 4
```

Output for the Sample Input 2

```
42
```

Sample Input 3

```
8
1 5 10
2 6 3
3 7 7
4 9 5
8 12 8
10 13 2
11 15 6
14 16 9
```

Output for the Sample Input 3

```
38
```

Sample Input 4

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

Output for the Sample Input 4

```
5
```

Problem K

Square

Time Limit: 1 Second

Given an integer n , write a program to determine whether there exists a square of area n whose four vertices are all integer lattice points in the two-dimensional plane.

Input

Your program is to read from standard input. The input is a line containing an integer n ($2 \leq n \leq 1,000,000$).

Output

Your program is to write to standard output. If there exists a square of area n whose four vertices are all integer lattice points in the two-dimensional plane, print YES; otherwise, print NO.

The following shows the sample input and output for four test cases.

Sample Input 1	Output for the Sample Input 1
2	YES
Sample Input 2	Output for the Sample Input 2
3	NO
Sample Input 3	Output for the Sample Input 3
500	YES
Sample Input 4	Output for the Sample Input 4
99999	NO

Problem K

Square

제한 시간: 1 초

정수 n 에 대해, 네 꼭짓점이 모두 2차원 정수 격자점이고 넓이가 n 인 정사각형의 존재 여부를 판단하는 프로그램을 작성하시오.

Input

입력은 표준 입력을 사용한다. 첫 번째 줄에 정수 n ($2 \leq n \leq 1,000,000$)이 주어진다.

Output

출력은 표준 출력을 사용한다. 입력에 대해, 네 꼭짓점이 모두 2차원 정수 격자점이고 넓이가 n 인 정사각형이 존재한다면 YES, 존재하지 않는다면 NO 를 출력하시오.

다음은 네 테스트 케이스에 대한 입출력 예이다.

Sample Input 1	Output for the Sample Input 1
2	YES
Sample Input 2	Output for the Sample Input 2
3	NO
Sample Input 3	Output for the Sample Input 3
500	YES
Sample Input 4	Output for the Sample Input 4
99999	NO