Problem A. Coloring Roads

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	1024 mebibytes

In RUN-land, there are n cities numbered 1 to n. Some pairs of cities are connected by a bidirectional road. It happens that there are $n - 1$ roads in total and that for any two cities, and there is a unique path from one to the other.

The city number 1 is the capital. Initially all roads have no color. Alex, the king of RUN-land asks you to perform the following query Q times.

- `u c m`: Given a city u, a color c, and an integer m, color all the roads on the unique path from u to the capital in the color c. Even if a road already has a color, change its color to c. After coloring, compute the number of colors in which exactly m roads are colored.

Given Q queries in total, compute the answer for the second part of each query.

Input

The first line of the input contains three integers n ,C, Q ($1 \le n, C, Q \le 2 \times 10^5$), separated by a single space, which are the number of cities in RUN-land, the number of possible colors, and the number of queries, respectively. Each of the next $n - 1$ lines contains two integers u, v ($1 \le u, v \le n$) meaning that there is a bidirectional road directly connecting the cities numbered u and v.

Each of the next Q lines contains a query, which contains 3 integers u, c, m as described in the statement. ($1 \le u \le n$, $1 \le c \le C$, $0 \le m \le n - 1$)

Output

Print Q lines, one for each query. Each line must contain one integer, the answer to the corresponding query.

Example

standard input	standard output
6 5 5 1 3 2 3 1 4 6 3 5 2 5 1 3 6 2 2 2 3 1 4 4 1 1 5 0	1 2 2 3 1

Note

The answer for the last query is 1 since color 5 is used in 0 roads.

Problem B. Dev, Please Add This!

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

Jaemin developed a puzzle game app.

A ball is placed on a cell in a grid. You can roll the ball upwards, downwards, to the left, or to the right. The ball rolls in that direction until it hits a wall or the boundary. Some cells have stars on them, and the player obtains the star when the ball stops there or passes through it. The objective of the game is to obtain all stars in the grid.

This app has a level editor where players can make and share their own levels. One day, someone suggested: ¡¡Please add a feature to check if my level is possible to solve!¿¿ Easier said than done, right?

Input

The first line of the input consists of two integers, H (height) and W (width). ($1 \le H, W \le 50$) Each of the next H lines contains a string of legnth W which describes each row of the grid. '`#`' means a wall, '`.`' means a blank space, '`O`' means a ball, and '`*`' means a star. There is exactly one ball and at least one star on the grid.

Output

Print "`YES`" if it is possible to obtain all stars, otherwise "`NO`".

Examples

standard input	standard output
3 7 #..O..# #.###.# *..#..*	NO
6 6 *..*## ..O... *..*#. ####*. ....#	YES

Problem C. Dstorv

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

In a straight line, there is *hand objects* and *flower objects* moving in consistent speed. All hand objects move toward the left, and all flower objects move toward the right. The total number of objects are N.

As the speed is consistent, no object of same type will collide to each other. However, there is a chance that the flower and hand might collide. In such case, one of them disappears, and other continues to move in the same direction in speed.

Which one to be removed, is determined by a probability. Flower object disappears with probability $\frac{r}{r+h}$. Hand object disappears with probability $\frac{h}{r+h}$. r, h are the integers given as an input.

After a sufficient amount of time, there will be no collision, as hands would've all gone left, and flowers would've all gone right. Your task is to calculate the probability that there is exactly A flower object and B hand object remaining after all the collision.

Input

The first line of the input contains three integers N, r, h. $(1 \le N \le 5\,000, 0 \le r, h \le 1\,000\,000, 1 \le r + h)$

The next line contains a string of length N consisting of two character `R`, `H`. This denotes the initial placement of objects in left-to-right order. If a character is `R`, it denotes a flower object. If a character is `H`, it denotes a hand object.

The last line contains two integers A, B. $(0 \le A, B \le N, 1 \le A + B \le N)$

Output

It can be shown that the answer can be represented as $\frac{P}{Q}$, where P, Q are coprime integer and $Q \neq 0 \pmod{10^9 + 7}$.

Output the value of $P \times Q^{-1} \bmod 10^9 + 7$.

Example

standard input	standard output
4 1 1 RHRH 1 1	250000002

Note

The answer for the sample is $\frac{1}{4}$.

Problem D. Dumae

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

Do you know *Dumae*? It is a nickname of the most famous restaurant nearby KAIST, *Dumae Charcoal-grilled Barbecue*. Because *Dumae* is a very famous restaurant, lots of KAIST students stand in line even though it has not opened yet. Students wonder how long they have to wait, so they started to guess their order.

There are N students in waiting line and each of them has a distinct student ID from 1 to N. Student i (student with student ID i) guessed that he/she is either L_i-th, $(L_i + 1)$-th, ..., $(R_i - 1)$-th, or R_i-th person in the line. (i.e. the number of people standing relatively in front of him/her is in the interval $[L_i - 1, R_i - 1]$) Also, M claims are made, of which the i-th says that student v_i can see student u_i in the waiting line. It means student u_i is relatively in front of student v_i.

You wonder if all of students' guesses and claims were right. Find an order of waiting line that satisfies all the guesses and claims, or report that such an order does not exist.

Input

The first line contains two space-separated integers N, M. $(1 \le N \le 300\,000, 0 \le M \le 1\,000\,000)$

In the next N lines, two space-separated integers L_i, R_i are given. $(1 \le L_i \le R_i \le N)$

In the next M lines, two space-separated integers u_i, v_i are given. $(1 \le u_i \le N,\ 1 \le v_i \le N,\ u_i \ne v_i)$

Output

If there is no answer that satisfies the condition, print -1.

Otherwise, print N lines. In the i-th line, print the student ID of the i-th student from the front.

Examples

standard input	standard output
3 3 1 3 1 3 1 3 1 2 2 3 3 1	-1
3 3 1 3 1 3 1 3 1 2 2 3 1 3	1 2 3

Problem E. Electronic Circuit

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

Joon is taking General Physics II and he is now studying electronic circuits. An electronic circuit consists of several nodes and undirected wires each connecting two distinct nodes. Moreover, a circuit has two distinctive end nodes; a source node and a sink node, where a voltage is applied (usually it is applied by additional wire with a battery connecting the two nodes, but we will neglect it). Each wire has a resistance, and Joon should know how to calculate the composite resistance of a circuit.

By the way, Joon hates complicated things. So he only cares about circuits that can be made by series and parallel compositions, since they are easy to calculate the composite resistance. He calls them 'nice' circuits; formally, a nice circuit can be defined as follows.

- A circuit with a single wire connecting two end nodes is nice.
- A circuit obtained by merging the sink node of a nice circuit C_1 and the source node of a nice circuit C_2 into a single node is nice. The source node and the sink node of the obtained circuit are the source node of C_1 and the sink node of C_2, respectively.
- A circuit obtained by merging the two source nodes of nice circuits C_1 and C_2 into a single node, and merging the two sink nodes of C_1 and C_2 into a single node, is nice. The two end nodes of the obtained circuit are the respective merged end nodes.

He made a circuit with his wires to calculate the composite resistance, but his friend Pringles screwed up his circuit, so now Joon does not know what the end nodes are. To make things worse, he is not even sure whether the circuit is nice or not.

Joon will give you the circuit. He kindly asks you whether the circuit can be nice by appropriately choosing two end nodes. Be careful that there may be multiple wires connecting two nodes.

Input

The first line contains two integers, n and m ($2 \le n \le 10^5$, $1 \le m \le 3 \times 10^5$), where n is the number of nodes and m is the number of wires. All nodes are numbered from 1 to n.

In the following m lines, each line contains two integers u and v ($1 \le u, v \le n$, $u \ne v$), which represents a wire connecting u and v. It is guaranteed that every node is attached to at least one wire; otherwise the node does not exist!

Output

Print "`Yes`" if the given circuit can be nice, or "`No`" otherwise.

Examples

standard input	standard output
4 6 1 2 2 3 2 3 3 4 1 4 1 4	Yes
4 6 1 2 1 3 1 4 2 3 2 4 3 4	No
9 12 1 9 1 4 5 4 6 5 1 5 8 1 3 6 6 8 3 8 2 9 9 7 7 2	Yes

Problem F. Fake Plastic Trees

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

Tree is a recursive structure, which is either:

- **Empty**. Empty tree is denoted as -1 and has a size of 0.
- **Non-empty**. Non-empty tree T is denoted as a pair of two trees (T_1, T_2), where T_1 is called **left subtree** of T, and T_2 is called **right subtree** of T. If $T = (-1, -1)$, then we call such T a **leaf**. Leaf has a size of 1, and non-leaf has a size of $|T_1| + |T_2|$, where $|T_1|$ is the size of T_1, and $|T_2|$ is the size of T_2.

A non-empty tree T is a **Fake Plastic Tree**, if the tree is *balanced*. Formally, Let $T = (T_1, T_2)$. If $|T_1| = |T_2|$ or $|T_1| = |T_2| + 1$ holds, then T is a Fake Plastic Tree.

In computer science, trees are commonly used as a data structure, and they are stored in a memory. At first, there are no trees in the memory, and only an imaginary *null pointer* exists (which corresponds to empty tree, -1). You can allocate a tree in the memory, by setting T_1 and T_2 as either a null pointer or a pointer of an existing tree. Then, the memory is extended by adding $T = (T_1, T_2)$ into its structure. Note that pointer can be described as a small integer, reducing the need for explicitly storing the whole tree.

Formally, memory M is an inductive structure, which at first contains only empty tree -1. ($M = \{-1\}$). You can expand the memory with following operation $M \leftarrow M \cup \{(T_1, T_2)\}$, where $T_1 \in M, T_2 \in M$. If a tree T is inserted in i-th stage, then it has the **index** $i - 1$. For a tree with index i, their subtrees can be represented as a pair of integer in range $[-1, i - 1]$.

Your task is to construct a memory M, which satisfies the following :

- Every tree in M is either empty or Fake Plastic Tree.
- M has at most 125 non-empty trees.
- There exists a tree $T \in M$, where $|T| = N$ holds. N is an integer, and is given as an input.

Input

The first line contains a single integer T, the number of test cases. ($1 \le T \le 2000$)

In the next T lines, a single integer N is given, which indicates the number of leaves your tree should contain. ($1 \le N \le 10^{18}$)

Output

For each case, you should print $V + 2$ lines, where V is the number of non-empty trees in M. ($1 \le V \le 125$).

In the first line, you should print single integer V.

In the next V lines, you should print two space-separated integer L_i, R_i, which indicates the index of left subtree and right subtree for a tree with index i. ($-1 \le L_i, R_i \le i - 1$).

In the $V + 2$-th line, you should print P, the index of the tree which contains N nodes. ($0 \le P \le V - 1$).

It is guaranteed that an answer always exists under the given condition.

Example

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

Problem G. Fascination Street

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

A street named *Fascination Street* is divided into N equal length of blocks. For each block i ($1 \le i \le N$), it has block $i-1$ in its left side if $i > 1$, and block $i+1$ in its right side if $i < N$.

Unlike its name, the street is infamous to be a dark and eerie place in the night. To solve this, Robert decided to install the streetlight for some of the blocks. The cost of installing a streetlight for i-th block is W_i, and the total cost is the sum of each installation cost. After installing, every block should either have a streetlight, or have a streetlight in it's left or right block.

Robert also has some tricks to reduce the cost. Before installing the streetlight, Robert selects two distinct blocks i and j, and exchange their position. After the operation, the cost of installation is exchanged. In other words, the operation simply swaps the value of W_i and W_j. This operation have no cost, but Robert can only perform it at most K times.

Now, given the array W and the maximum possible number of operations K, you should find the minimum cost of lighting the whole street.

Input

The first line contains two space-separated integers N, K. N is the number of blocks, and K is the maximum possible number of operations. ($1 \le N \le 250000, 0 \le K \le 9$)

The second line contains N space-separated integers $W_1, W_2 \cdots W_N$, where W_i is the cost of installing a streetlight for i-th block. ($0 \le W_i \le 10^9$)

Output

Print a single integer which contains the minimum cost of lighting the whole street.

Examples

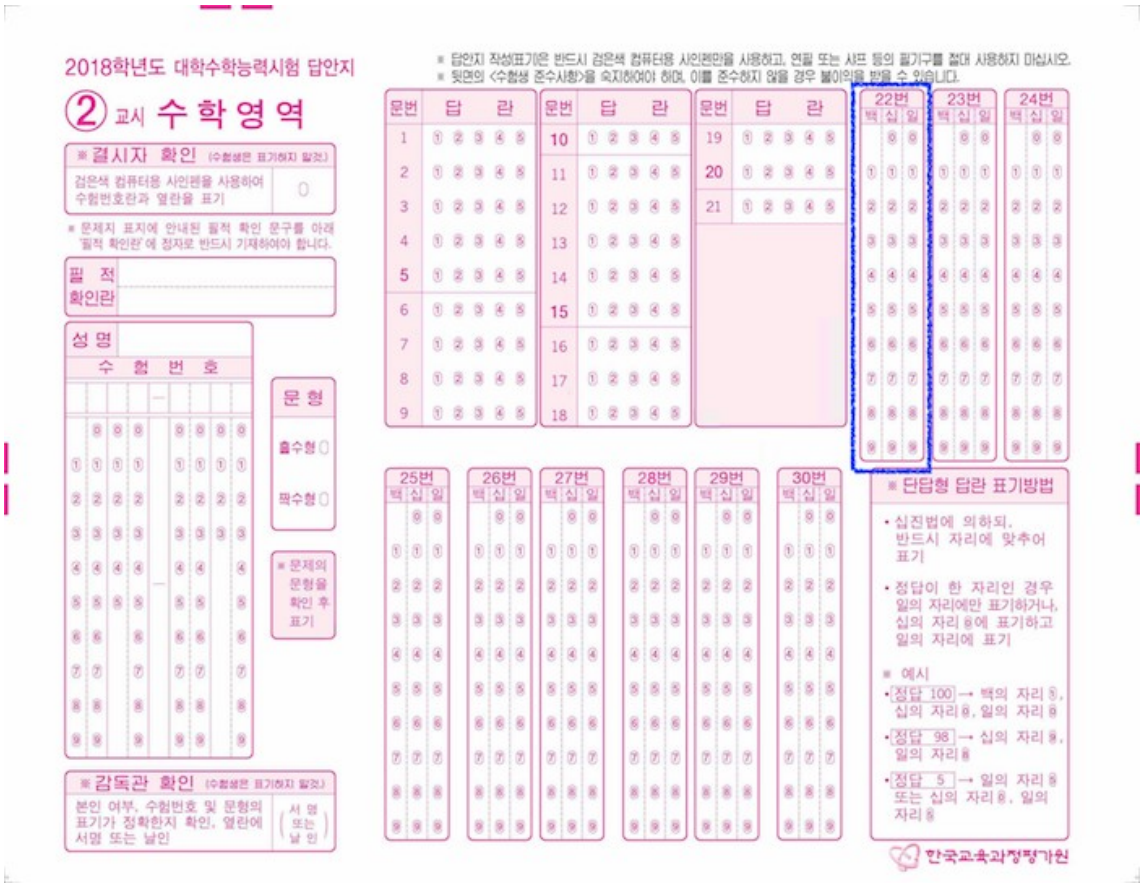
standard input	standard output
5 0 1 3 10 3 1	4
5 1 1 3 10 3 1	2
12 0 317 448 258 208 284 248 315 367 562 500 426 390	1525
12 2 317 448 258 208 284 248 315 367 562 500 426 390	1107

Problem H. Fractions

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

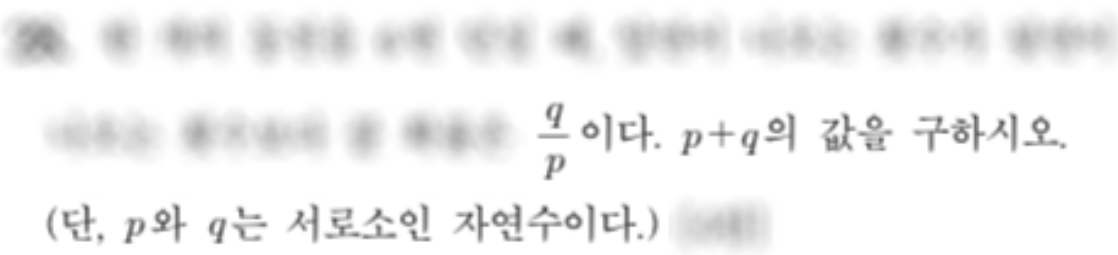
About 44 days are left before *College Scholastic Ability Test* is held. This exam aims to measure students' achievement of National Curriculum standards and scholastic ability required for college education. (`http://www.kice.re.kr/sub/info.do?m=0205&s=english`)

One of the subjects covered by this test is Mathematics, which consists of 21 multiple choice questions and 9 short-answer questions. The answer of each short-answer question is guaranteed to be a *unique positive integer below 1 000*, as you can see from the answer sheet below.



However, the organizers might want to give students short-answer questions with non-integer answers, such as $2\sqrt{3}$ or $\frac{5}{3}$. Usually, the workaround is to write the answer in a canonical form, and then sum up all the integers inside that form and ask students to write that number instead.

In particular, when the answer is a positive rational number $\frac{a}{b}$, the organizers usually ask students to reduce it and sum up the numerator and the denominator of the reduced fraction. For example, when the answer is $\frac{18}{10}$, the student should reduce it to $\frac{9}{5}$ and write the final answer as $9 + 5 = 14$.



$\frac{q}{p}$이다. $p+q$의 값을 구하시오.

(단, p와 q는 서로소인 자연수이다.)

However, when the answer is $\frac{521}{500}$, the reduced fraction is also $\frac{521}{500}$, so the student should write the final answer as $521 + 500 = 1021$. But this shouldn't happen, since all the answers for the short-answer questions are below 1 000. To avoid this situation, the organizers should make sure that after reducing the fraction, the sum of the numerator and the denominator shouldn't exceed 999. Let's call such fractions as *Suneung Fractions*. For example, $\frac{1996}{2}$ and $\frac{18}{10}$ are *Suneung fractions*, while $\frac{1998}{2}$ and $\frac{521}{500}$ are not.

Suppose that, this year, one of the organizers wrote a problem, and the answer to that problem is $\frac{x}{y}$. Since the problem is not finalized yet, the only thing we know is $A \le x \le B$ and $C \le y \le D$ holds, for given A, B, C, D. The organizers want to know, among all the pairs (x, y), how many of $\frac{x}{y}$ is a *Suneung fraction*. Write a program that counts this number.

Input

The first and only line contains four space-separated integers A, B, C and D ($1 \le A \le B \le 10^{12}$, $1 \le C \le D \le 10^{12}$)

Output

Print the number of integral pairs (x, y) ($A \le x \le B$, $C \le y \le D$), where $\frac{x}{y}$ is a *Suneung fraction*.

Examples

standard input	standard output
5 8 3 6	16
2018 2019 2018 2019	2

Problem I. Game on Plane

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

You are given N points on a plane. These points are precisely the set of vertices of some regular N-gon. Koosaga, an extreme villain, is challenging you with a game using these points. You and Koosaga alternatively take turns, and in each turn, the player

1. chooses two of the given points, then
2. draws the line segment connecting the two chosen points.

Also, the newly drawn line segment must not intersect with any of the previously drawn line segments in the interior. It is possible for two segments to meet at their endpoints. If at any point of the game, there exists a convex polygon consisting of the drawn line segments, the game ends and the last player who made the move wins.

Given the integer N, Koosaga is letting you decide who will move first. Your task is decide whether you need to move first or the second so that you can win regardless of Koosaga's moves.

Input

The input consists of many test cases. The first line contains an integer T ($1 \le T \le 5000$), the number of test cases. Each of the following T test cases is consisted of one line containing the integer N ($3 \le N \le 5000$).

Output

For each test case, print one line containing the string "`First`" if you need to move first or "`Second`" if you need to move second so that you can win regardless of Koosaga's moves.

Example

standard input	standard output
2 3 5	First Second

Problem J. Histogram Sequence

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

A histogram is a polygon made by aligning N adjacent rectangles that share a common base line. Each rectangle is called a *bar*. The i-th bar from the left has width 1 and height H_i.

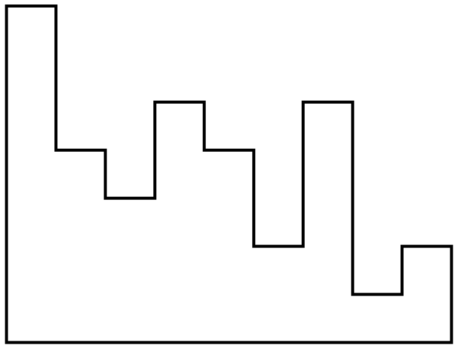


Figure: This picture depicts a case when $N = 9$ and $H = [7, 4, 3, 5, 4, 2, 5, 1, 2]$.

One day, you wanted to find the area of the largest rectangle contained in the given histogram. What you did was to make a list of integers A by the following procedure:

- For each $1 \le i \le j \le N$, calculate the largest area of the rectangle contained in the histogram, where the rectangle's base line coincides with the base line of the $i, i+1, \cdots, j-1, j$-th bar. Add the area to the list A.

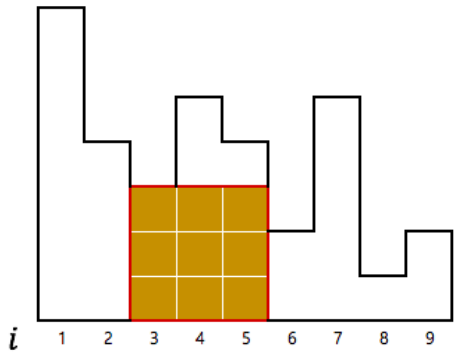


Figure: This picture depicts a case when $i = 3$ and $j = 5$. The area is 9.

The length of the list A is exactly $\frac{N(N+1)}{2}$ since you chose each pair (i, j) exactly once. To make your life easier, you sorted the list A in non-decreasing order. Now, to find the largest area of the rectangle contained in the histogram, you just need to read the last element of A, $A_{N(N+1)/2}$.

However, you are not satisfied with this at all, so I decided to let you compute some part of the list A. You have to write a program that, given two indices L and R ($L \le R$), calculate the values $A_{L..R}$, i.e. $A_L, A_{L+1}, \cdots, A_{R-1}, A_R$.

Input

The first line of the input contains an integer N ($1 \le N \le 300\,000$) which is the number of bars in the histogram.

The next line contains N space-separated positive integers $H_1, H_2, \cdots, H_N$ ($1 \le H_i \le 10^9$), where H_i is the height of the i-th bar.

The last line contains two integers L and R ($1 \le L \le R \le \frac{N(N+1)}{2}$, $R - L + 1 \le 300\,000$).

Output

Print $R - L + 1$ integers. The j-th ($1 \le j \le R - L + 1$) of them should be the $(L + j - 1)$-th element of the list A, i.e. A_{L+j-1}.

Example

standard input	standard output
9 7 4 3 5 4 2 5 1 2 42 45	12 12 14 15

Problem K. Interesting Drug

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

During Younghun's trip to IOI in Japan, he was nervous with the upcoming competition. So, he went out for nearby playground to relax himself. In the playground, there was N drugs lying in a straight line. The drugs are numbered from 1 to N from left to right, and their locations are distinct. Younghun was curious about the drug, and swallowed one of them.

After then, due to the effect of drug, Younghun started to dance unconsciously. Younghun was unable to control himself from dancing, but was only able to control the direction he was moving (left or right in the straight line). When Younghun is moving, he automatically swallows any drug in his way. After swallowing the drug, the drug disappears from the playground.

In the end, Younghun swallowed all N drugs, and found his consciousness. He was not shocked about what he've done (just like all great K-pop stars, dancing is his life), but was worried about the potential health risks of drugs. Let $P = \{P_1, P_2, \cdots, P_n\}$ be the sequence of drugs in order of Younghun's consumption. After some research, Younghun realized that he receives D_i damage if $P_{C_i} = i$.

Write a program which calculates the maximum possible damage dealt by starting at drug i. In other words, for each $1 \le i \le N$, you should calculate the maximum possible damage among all possible sequence of consumption with $P_1 = i$.

Input

The first line of the input contains a single integer N, denoting the number of drugs. ($2 \le N \le 300\,000$)

The next line contains N space-separated integer $C_1, C_2, \cdots, C_N$, which is the sequence C as explained above. ($1 \le C_i \le N$)

The last line contains N space-separated integer $D_1, D_2, \cdots, D_N$, which is the possible damage dealt by drug i. ($1 \le D_i \le 10^9$)

Output

You should print a single line containing N space-separated integer: The i-th number should be the maxmum possibly damage dealt by starting at drug i.

Example

standard input	standard output
5 1 3 3 1 3 4 3 1 2 10	5 1 10 12 1

Note

If $i = 1$, Younghun can only move right. This gives $P = \{1, 2, 3, 4, 5\}$, which deals damage $1 + 4 = 5$.

If $i = 2$, Younghun can move left, right, right, right. This gives $P = \{2, 1, 3, 4, 5\}$, which deals damage 1.

If $i = 3$, Younghun can move right, right, left, left. This gives $P = \{3, 4, 5, 2, 1\}$, which deals damage 10.

If $i = 4$, Younghun can move left, right, left, left. This gives $P = \{4, 3, 5, 2, 1\}$, which deals damage $2 + 10 = 12$.

If $i = 5$, Younghun can only move left. This gives $P = \{5, 4, 3, 2, 1\}$, which deals damage 1.

Problem L. Timsort

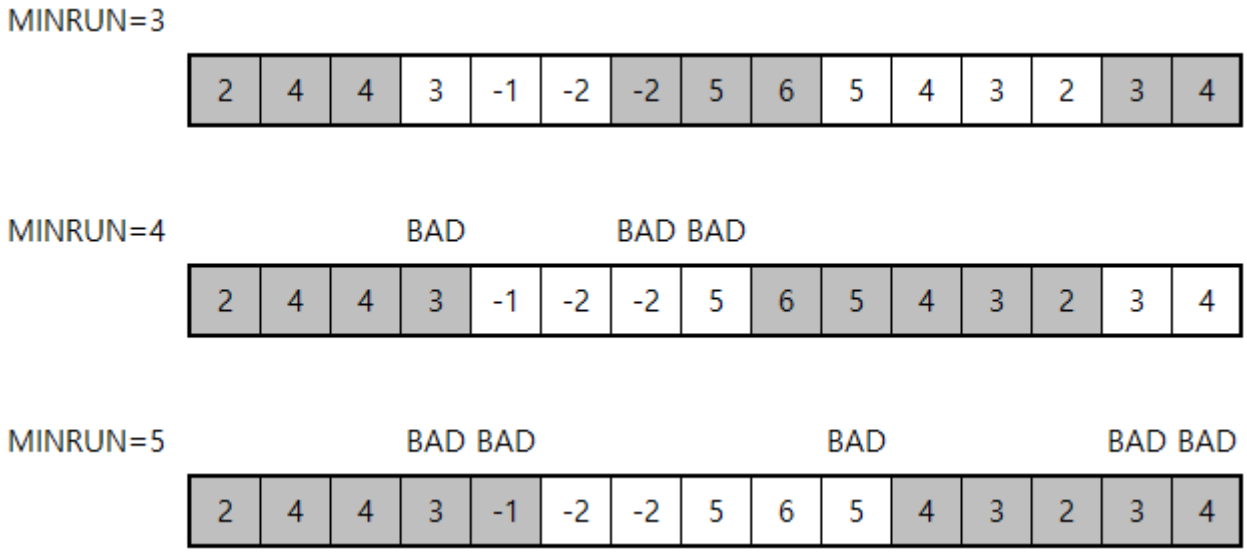
Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

Python uses a sorting algorithm called Timsort. In short, the algorithm splits the list into mostly sorted subarrays, sorts each subarray using insertion sort, and merges the sorted subarrays. If the list is already mostly sorted, this algorithm runs very quickly.

In this problem, let's focus on the splitting part:

1. Take the longest non-decreasing ($a_0 \le a_1 \le a_2 \le ...$) or strictly decreasing ($a_0 > a_1 > a_2 > ...$) subarray that starts from the current index.

2. If the length of the subarray is less than $MINRUN$, take more elements until the length equals $MINRUN$ or all elements are taken. Let's call these additional elements "bad elements".



If $MINRUN$ is too small, there are too many subarrays to merge; if $MINRUN$ is too large, it takes too much time to apply insertion sort. For this reason, it is important to set an appropriate value of $MINRUN$. Also, $N/MINRUN$ should be close to a power of 2 to ensure a balanced merging, but we will not consider this in this problem. For each $MINRUN$ value, find the number of subarrays and bad elements.

Input

The first line consists of a single integer $N (5 \le N \le 100,000)$, the length of the array. The second line consists of N integers, the elements of the array. The absolute value of each element is at most 10^9. The third line consists of a single integer Q $(1 \le Q \le 100,000)$, the number of queries. Each of the next Q lines contains a single integer $MINRUN$ $(2 \le MINRUN \le N)$.

Output

For each query, print the number of subarrays and bad elements in one line.

Example

standard input	standard output
15 2 4 4 3 -1 -2 -2 5 6 5 4 3 2 3 4 3 3 4 5	5 0 4 3 3 5

Problem M. Utilitarianism

Input file:	*standard input*
Output file:	*standard output*
Time limit:	8 seconds
Memory limit:	1024 mebibytes

In RUN-land, there are n cities numbered 1 to n. Some pairs of cities are connected by a bidirectional road. It happens that there are $n - 1$ roads in total and that for any two cities, there is a unique path from one to the other. Also, each road is assigned an integer called the *value*.

Today, to honor the k co-founders of RUN-land, Alex, the king of RUN-land, has decided to choose k different roads and give one road to each of the k co-founders. To prevent unnecessary conflicts, there should be no city that is connected to more than one chosen roads.

In this process, Alex, the king of RUN-land, does not care about who gets what road. Instead, Alex, the king of RUN-land, is only interested in the sum of the values of the k chosen roads. Your task is to choose the roads to maximize this sum.

Input

The first line contains two integers n and k ($2 \le n \le 250\,000$, $1 \le k \le n - 1$), which are the number of cities in RUN-land, and the number of roads to choose. Each of the next $n - 1$ lines contains three integers u, v, c ($1 \le u, v \le n$, $-10^6 \le c \le 10^6$), which means that the city u and the city v are directly connected with a bidirectional road with value c.

Output

If there is no way to choose k roads to satisfy the conditions, print "`Impossible`". Otherwise, print one integer, the maximum sum of the values of the k chosen roads.

Examples

standard input	standard output
5 1 1 2 2 2 3 3 2 4 10 4 5 6	10
5 2 1 2 2 2 3 3 2 4 10 4 5 6	9
5 3 1 2 2 2 3 3 2 4 10 4 5 6	Impossible