# A. Mahmoud and Ehab and the bipartiteness

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Mahmoud and Ehab continue their adventures! As everybody in the evil land knows, Dr. Evil likes bipartite graphs, especially trees.

A tree is a connected acyclic graph. A bipartite graph is a graph, whose vertices can be partitioned into $2$ sets in such a way, that for each edge $(u, v)$ that belongs to the graph, $u$ and $v$ belong to different sets. You can find more formal definitions of a tree and a bipartite graph in the notes section below.

Dr. Evil gave Mahmoud and Ehab a tree consisting of $n$ nodes and asked them to add edges to it in such a way, that the graph is still bipartite. Besides, after adding these edges the graph should be simple (doesn't contain loops or multiple edges). What is the maximum number of edges they can add?

A loop is an edge, which connects a node with itself. Graph doesn't contain multiple edges when for each pair of nodes there is no more than one edge between them. **A cycle and a loop aren't the same** .

## Input

The first line of input contains an integer $n$ — the number of nodes in the tree ($1 \le n \le 10^5$).

The next $n - 1$ lines contain integers $u$ and $v$ ($1 \le u, v \le n, u \ne v$) — the description of the edges of the tree.

It's guaranteed that the given graph is a tree.

## Output

Output one integer — the maximum number of edges that Mahmoud and Ehab can add to the tree while fulfilling the conditions.

### Examples

| input |
| --- |
| 3<br>1 2<br>1 3 |
| **output** |
| 0 |

| input |
| --- |
| 5<br>1 2<br>2 3<br>3 4<br>4 5 |
| **output** |
| 2 |

## Note

Tree definition: https://en.wikipedia.org/wiki/Tree_(graph_theory)

Bipartite graph definition: https://en.wikipedia.org/wiki/Bipartite_graph

In the first test case the only edge that can be added in such a way, that graph won't contain loops or multiple edges is $(2, 3)$, but adding this edge will make the graph non-bipartite so the answer is 0.

In the second test case Mahmoud and Ehab can add edges $(1, 4)$ and $(2, 5)$.

# B. Combination

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Ilya plays a card game by the following rules.

A player has several cards. Each card contains two non-negative integers inscribed, one at the top of the card and one at the bottom. At the beginning of the round the player chooses one of his cards to play it. If the top of the card contains number $a_i$, and the bottom contains number $b_i$, then when the player is playing the card, he gets $a_i$ points and also gets the opportunity to play additional $b_i$ cards. After the playing the card is discarded.

More formally: let's say that there is a counter of the cards that can be played. At the beginning of the round the counter equals one. When a card is played, the counter decreases by one for the played card and increases by the number $b_i$, which is written at the bottom of the card. Then the played card is discarded. If after that the counter is not equal to zero, the player gets the opportunity to play another card from the remaining cards. The round ends when the counter reaches zero or the player runs out of cards.

Of course, Ilya wants to get as many points as possible. Can you determine the maximum number of points he can score provided that you know his cards?

## Input

The first line contains a single integer $n$ ($1 \le n \le 1000$) — the number of cards Ilya has.

Each of the next $n$ lines contains two non-negative space-separated integers — $a_i$ and $b_i$ ($0 \le a_i, b_i \le 10^4$) — the numbers, written at the top and the bottom of the $i$-th card correspondingly.

## Output

Print the single number — the maximum number of points you can score in one round by the described rules.

### Examples

| input |
| --- |
| 2<br>1 0<br>2 0 |
| **output** |
| 2 |

| input |
| --- |
| 3<br>1 0<br>2 0<br>0 2 |
| **output** |
| 3 |

## Note

In the first sample none of two cards brings extra moves, so you should play the one that will bring more points.

In the second sample you should first play the third card that doesn't bring any points but lets you play both remaining cards.

# C. Producing Snow

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Alice likes snow a lot! Unfortunately, this year's winter is already over, and she can't expect to have any more of it. Bob has thus bought her a gift — a large snow maker. He plans to make some amount of snow every day. On day $i$ he will make a pile of snow of volume $V_i$ and put it in her garden.

Each day, every pile will shrink a little due to melting. More precisely, when the temperature on a given day is $T_i$, each pile will reduce its volume by $T_i$. If this would reduce the volume of a pile to or below zero, it disappears forever. All snow piles are independent of each other.

Note that the pile made on day $i$ already loses part of its volume on the same day. In an extreme case, this may mean that there are no piles left at the end of a particular day.

You are given the initial pile sizes and the temperature on each day. Determine the total volume of snow melted on each day.

## Input

The first line contains a single integer $N$ ($1 \le N \le 10^5$) — the number of days.

The second line contains $N$ integers $V_1, V_2, ..., V_N$ ($0 \le V_i \le 10^9$), where $V_i$ is the initial size of a snow pile made on the day $i$.

The third line contains $N$ integers $T_1, T_2, ..., T_N$ ($0 \le T_i \le 10^9$), where $T_i$ is the temperature on the day $i$.

## Output

Output a single line with $N$ integers, where the $i$-th integer represents the total volume of snow melted on day $i$.

### Examples

| input |
| --- |
| 3<br>10 10 5<br>5 7 2 |
| **output** |
| 5 12 4 |

| input |
| --- |
| 5<br>30 25 20 15 10<br>9 10 12 4 13 |
| **output** |
| 9 20 35 11 25 |

## Note

In the first sample, Bob first makes a snow pile of volume 10, which melts to the size of 5 on the same day. On the second day, he makes another pile of size 10. Since it is a bit warmer than the day before, the first pile disappears completely while the second pile shrinks to 3. At the end of the second day, he has only a single pile of size 3. On the third day he makes a smaller pile than usual, but as the temperature dropped too, both piles survive till the end of the day.

# D. Password

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Asterix, Obelix and their temporary buddies Suffix and Prefix has finally found the Harmony temple. However, its doors were firmly locked and even Obelix had no luck opening them.

A little later they found a string $s$, carved on a rock below the temple's gates. Asterix supposed that that's the password that opens the temple and read the string aloud. However, nothing happened. Then Asterix supposed that a password is some substring $t$ of the string $s$.

Prefix supposed that the substring $t$ is the beginning of the string $s$; Suffix supposed that the substring $t$ should be the end of the string $s$; and Obelix supposed that $t$ should be located somewhere inside the string $s$, that is, $t$ is neither its beginning, nor its end.

Asterix chose the substring $t$ so as to please all his companions. Besides, from all acceptable variants Asterix chose the longest one (as Asterix loves long strings). When Asterix read the substring $t$ aloud, the temple doors opened.

You know the string $s$. Find the substring $t$ or determine that such substring does not exist and all that's been written above is just a nice legend.

## Input

You are given the string $s$ whose length can vary from $1$ to $10^6$ (inclusive), consisting of small Latin letters.

## Output

Print the string $t$. If a suitable $t$ string does not exist, then print "`Just a legend`" without the quotes.

### Examples

| input |
| --- |
| fixprefixsuffix |
| **output** |
| fix |

| input |
| --- |
| abcdabc |
| **output** |
| Just a legend |

# E. MUH and Cube Walls

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polar bears Menshykov and Uslada from the zoo of St. Petersburg and elephant Horace from the zoo of Kiev got hold of lots of wooden cubes somewhere. They started making cube towers by placing the cubes one on top of the other. They defined multiple towers standing in a line as a wall. A wall can consist of towers of different heights.

Horace was the first to finish making his wall. He called his wall an elephant. The wall consists of $w$ towers. The bears also finished making their wall but they didn't give it a name. Their wall consists of $n$ towers. Horace looked at the bears' tower and wondered: in how many parts of the wall can he "see an elephant"? He can "see an elephant" on a segment of $w$ contiguous towers if the heights of the towers on the segment match as a sequence the heights of the towers in Horace's wall. In order to see as many elephants as possible, Horace can raise and lower his wall. He even can lower the wall below the ground level (see the pictures to the samples for clarification).

Your task is to count the number of segments where Horace can "see an elephant".

## Input

The first line contains two integers $n$ and $w$ ($1 \le n, w \le 2 \cdot 10^5$) — the number of towers in the bears' and the elephant's walls correspondingly. The second line contains $n$ integers $a_i$ ($1 \le a_i \le 10^9$) — the heights of the towers in the bears' wall. The third line contains $w$ integers $b_i$ ($1 \le b_i \le 10^9$) — the heights of the towers in the elephant's wall.

## Output

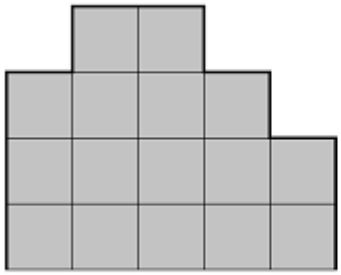
Print the number of segments in the bears' wall where Horace can "see an elephant".

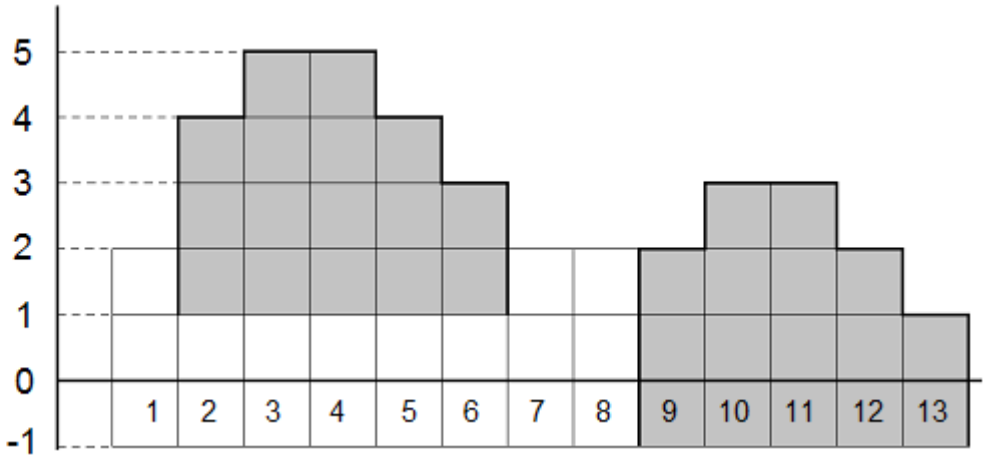
### Examples

| input |
| --- |
| 13 5<br>2 4 5 5 4 3 2 2 2 3 3 2 1<br>3 4 4 3 2 |
| **output** |
| 2 |

## Note

The picture to the left shows Horace's wall from the sample, the picture to the right shows the bears' wall. The segments where Horace can "see an elephant" are in gray.




5
4
3
2
1
0
-1
1
2
3
4
5
6
7
8
9
10
11
12
13

# F. Minimum path

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a matrix of size $n \times n$ filled with lowercase English letters. You can change no more than $k$ letters in this matrix.

Consider all paths from the upper left corner to the lower right corner that move from a cell to its neighboring cell to the right or down. Each path is associated with the string that is formed by all the letters in the cells the path visits. Thus, the length of each string is $2n - 1$.

Find the lexicographically smallest string that can be associated with a path after changing letters in at most $k$ cells of the matrix.

A string $a$ is lexicographically smaller than a string $b$, if the first different letter in $a$ and $b$ is smaller in $a$.

## Input

The first line contains two integers $n$ and $k$ ($1 \le n \le 2000$, $0 \le k \le n^2$) — the size of the matrix and the number of letters you can change.

Each of the next $n$ lines contains a string of $n$ lowercase English letters denoting one row of the matrix.

## Output

Output the lexicographically smallest string that can be associated with some valid path after changing no more than $k$ letters in the matrix.

### Examples

| input |
| --- |
| 4 2<br>abcd<br>bcde<br>bcad<br>bcde |
| **output** |
| aaabcde |

| input |
| --- |
| 5 3<br>bwwwz<br>hrhdh<br>sepsp<br>sqfaf<br>ajbvw |
| **output** |
| aaaepfafw |

| input |
| --- |
| 7 6<br>ypnxnnp<br>pnxonpm<br>nxanpou<br>xnnpmud |

```
nhtdudu
npmuduh
pmutsnz
```

**output**

```
aaaaaaadudsnz
```

### Note

In the first sample test case it is possible to change letters 'b' in cells $(2, 1)$ and $(3, 1)$ to 'a', then the minimum path contains cells $(1, 1), (2, 1), (3, 1), (4, 1), (4, 2), (4, 3), (4, 4)$. The first coordinate corresponds to the row and the second coordinate corresponds to the column.

# G. Segments

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given $n$ segments on the Ox-axis. You can drive a nail in any integer point on the Ox-axis line nail so, that all segments containing this point, are considered nailed down. If the nail passes through endpoint of some segment, this segment is considered to be nailed too. What is the smallest number of nails needed to nail all the segments down?

## Input

The first line of the input contains single integer number $n$ ($1 \le n \le 1000$) — amount of segments. Following $n$ lines contain descriptions of the segments. Each description is a pair of integer numbers — endpoints coordinates. All the coordinates don't exceed 10000 by absolute value. Segments can degenarate to points.

## Output

The first line should contain one integer number — the smallest number of nails needed to nail all the segments down. The second line should contain coordinates of driven nails separated by space in any order. If the answer is not unique, output any.

### Examples

| input |
| --- |
| 2<br>0 2<br>2 5 |

| output |
| --- |
| 1<br>2 |

| input |
| --- |
| 5<br>0 3<br>4 2<br>4 8<br>8 10<br>7 7 |

| output |
| --- |
| 3<br>7 10 3 |

# H. Thor

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Thor is getting used to the Earth. As a gift Loki gave him a smartphone. There are $n$ applications on this phone. Thor is fascinated by this phone. He has only one minor issue: he can't count the number of unread notifications generated by those applications (maybe Loki put a curse on it so he can't).

$q$ events are about to happen (in chronological order). They are of three types:

1. Application $x$ generates a notification (this new notification is unread).
2. Thor reads all notifications generated so far by application $x$ (he may re-read some notifications).
3. Thor reads the first $t$ notifications generated by phone applications (notifications generated in first $t$ events of the first type). It's guaranteed that there were at least $t$ events of the first type before this event. Please note that he doesn't read first $t$ unread notifications, he just reads the very first $t$ notifications generated on his phone and he may re-read some of them in this operation.

Please help Thor and tell him the number of unread notifications after each event. You may assume that initially there are no notifications in the phone.

## Input

The first line of input contains two integers $n$ and $q$ ($1 \le n, q \le 300\,000$) — the number of applications and the number of events to happen.

The next $q$ lines contain the events. The $i$-th of these lines starts with an integer $type_i$ — type of the $i$-th event. If $type_i = 1$ or $type_i = 2$ then it is followed by an integer $x_i$. Otherwise it is followed by an integer $t_i$ ($1 \le type_i \le 3, 1 \le x_i \le n, 1 \le t_i \le q$).

## Output

Print the number of unread notifications after each event.

### Examples

| input |
| --- |
| 3 4<br>1 3<br>1 1<br>1 2<br>2 3 |
| **output** |
| 1<br>2<br>3<br>2 |

| input |
| --- |
| 4 6<br>1 2<br>1 4<br>1 2<br>3 3<br>1 3<br>1 3 |

**output**

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

### Note

In the first sample:

1. Application $3$ generates a notification (there is $1$ unread notification).
2. Application $1$ generates a notification (there are $2$ unread notifications).
3. Application $2$ generates a notification (there are $3$ unread notifications).
4. Thor reads the notification generated by application $3$, there are $2$ unread notifications left.

In the second sample test:

1. Application $2$ generates a notification (there is $1$ unread notification).
2. Application $4$ generates a notification (there are $2$ unread notifications).
3. Application $2$ generates a notification (there are $3$ unread notifications).
4. Thor reads first three notifications and since there are only three of them so far, there will be no unread notification left.
5. Application $3$ generates a notification (there is $1$ unread notification).
6. Application $3$ generates a notification (there are $2$ unread notifications).

# I. Lucky Days

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Bob and Alice are often participating in various programming competitions. Like many competitive programmers, Alice and Bob have good and bad days. They noticed, that their lucky and unlucky days are repeating with some period. For example, for Alice days $[l_a; r_a]$ are lucky, then there are some unlucky days: $[r_a + 1; l_a + t_a - 1]$, and then there are lucky days again: $[l_a + t_a; r_a + t_a]$ and so on. In other words, the day is lucky for Alice if it lies in the segment $[l_a + kt_a; r_a + kt_a]$ for some non-negative integer $k$.

The Bob's lucky day have similar structure, however the parameters of his sequence are different: $l_b$, $r_b$, $t_b$. So a day is a lucky for Bob if it lies in a segment $[l_b + kt_b; r_b + kt_b]$, for some non-negative integer $k$.

Alice and Bob want to participate in team competitions together and so they want to find out what is the largest possible number of consecutive days, which are lucky for both Alice and Bob.

## Input

The first line contains three integers $l_a$, $r_a$, $t_a$ ($0 \le l_a \le r_a \le t_a - 1, 2 \le t_a \le 10^9$) and describes Alice's lucky days.

The second line contains three integers $l_b$, $r_b$, $t_b$ ($0 \le l_b \le r_b \le t_b - 1, 2 \le t_b \le 10^9$) and describes Bob's lucky days.

It is guaranteed that both Alice and Bob have some unlucky days.

## Output

Print one integer: the maximum number of days in the row that are lucky for both Alice and Bob.

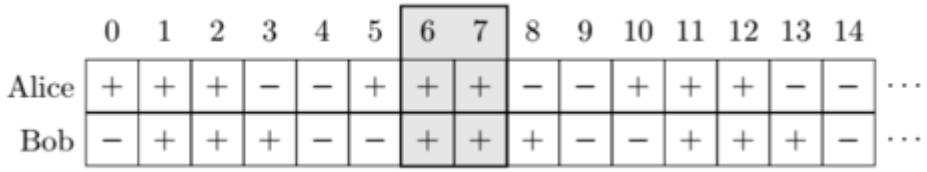
### Examples

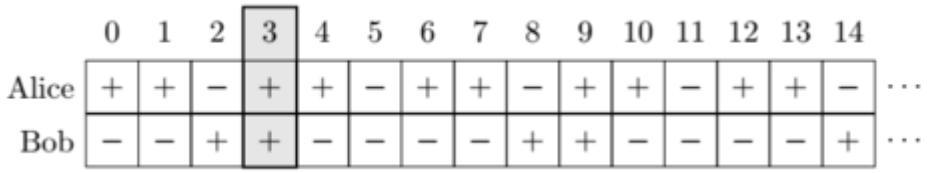
| input |
| --- |
| 0 2 5<br>1 3 5 |
| **output** |
| 2 |

| input |
| --- |
| 0 1 3<br>2 3 6 |
| **output** |
| 1 |

## Note

The graphs below correspond to the two sample tests and show the lucky and unlucky days of Alice and Bob as well as the possible solutions for these tests.


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
Alice + + + − − + + + − − + + + − − ···
Bob − + + + − − + + + − − + + + − ···


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
Alice + + − + + − + + − + + − + + − ···
Bob − − + + − − − − + + − − − − + ···

# J. Messenger

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Each employee of the "Blake Techologies" company uses a special messaging app "Blake Messenger". All the stuff likes this app and uses it constantly. However, some important futures are missing. For example, many users want to be able to search through the message history. It was already announced that the new feature will appear in the nearest update, when developers faced some troubles that only you may help them to solve.

All the messages are represented as a strings consisting of only lowercase English letters. In order to reduce the network load strings are represented in the special compressed form. Compression algorithm works as follows: string is represented as a concatenation of $n$ blocks, each block containing only equal characters. One block may be described as a pair $(l_i, c_i)$, where $l_i$ is the length of the $i$-th block and $c_i$ is the corresponding letter. Thus, the string $s$ may be written as the sequence of pairs $\langle (l_1, c_1), (l_2, c_2), \ldots, (l_n, c_n) \rangle$.

Your task is to write the program, that given two compressed string $t$ and $s$ finds all occurrences of $s$ in $t$. Developers know that there may be many such occurrences, so they only ask you to find the **number** of them. Note that $p$ is the starting position of some occurrence of $s$ in $t$ if and only if $t_p t_{p+1} \ldots t_{p+|s|-1} = s$, where $t_i$ is the $i$-th character of string $t$.

Note that the way to represent the string in compressed form may not be unique. For example string "aaaa" may be given as $\langle (4, a) \rangle$, $\langle (3, a), (1, a) \rangle$, $\langle (2, a), (2, a) \rangle$...

## Input

The first line of the input contains two integers $n$ and $m$ ($1 \le n, m \le 200\,000$) — the number of blocks in the strings $t$ and $s$, respectively.

The second line contains the descriptions of $n$ parts of string $t$ in the format "$l_i$-$c_i$" ($1 \le l_i \le 1\,000\,000$) — the length of the $i$-th part and the corresponding lowercase English letter.

The second line contains the descriptions of $m$ parts of string $s$ in the format "$l_i$-$c_i$" ($1 \le l_i \le 1\,000\,000$) — the length of the $i$-th part and the corresponding lowercase English letter.

## Output

Print a single integer — the number of occurrences of $s$ in $t$.

### Examples

| input |
| --- |
| 5 3<br>3-a 2-b 4-c 3-a 2-c<br>2-a 2-b 1-c |
| **output** |
| 1 |

| input |
| --- |
| 6 1<br>3-a 6-b 7-a 4-c 8-e 2-a<br>3-a |
| **output** |
| 6 |

| input |
| --- |
| 5 5<br>1-h 1-e 1-l 1-l 1-o<br>1-w 1-o 1-r 1-l 1-d |

| output |
| --- |
| 0 |

### Note

In the first sample, $t$ = "aaabbccccaaacc", and string $s$ = "aabbc". The only occurrence of string $s$ in string $t$ starts at position $p = 2$.

In the second sample, $t$ = "aaabbbbbbaaaaaaacccceeeeeeeeaa", and $s$ = "aaa". The occurrences of $s$ in $t$ start at positions $p = 1$, $p = 10$, $p = 11$, $p = 12$, $p = 13$ and $p = 14$.

# K. Party

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

$n$ people came to a party. Then those, who had no friends among people at the party, left. Then those, who had exactly 1 friend among those who stayed, left as well. Then those, who had exactly $2, 3, ..., n - 1$ friends among those who stayed by the moment of their leaving, did the same.

What is the maximum amount of people that could stay at the party in the end?

## Input

The first input line contains one number $t$ — amount of tests ($1 \le t \le 10^5$). Each of the following $t$ lines contains one integer number $n$ ($1 \le n \le 10^5$).

## Output

For each test output in a separate line one number — the maximum amount of people that could stay in the end.

### Examples

| input |
| --- |
| 1<br>3 |
| **output** |
| 1 |

# L. Perfect Security

time limit per test: 3.5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Alice has a very important message $M$ consisting of some non-negative integers that she wants to keep secret from Eve. Alice knows that the only theoretically secure cipher is one-time pad. Alice generates a random key $K$ of the length equal to the message's length. Alice computes the bitwise xor of each element of the message and the key ($A_i := M_i \oplus K_i$, where $\oplus$ denotes the bitwise XOR operation) and stores this encrypted message $A$. Alice is smart. Be like Alice.

For example, Alice may have wanted to store a message $M = (0, 15, 9, 18)$. She generated a key $K = (16, 7, 6, 3)$. The encrypted message is thus $A = (16, 8, 15, 17)$.

Alice realised that she cannot store the key with the encrypted message. Alice sent her key $K$ to Bob and deleted her own copy. Alice is smart. Really, be like Alice.

Bob realised that the encrypted message is only secure as long as the key is secret. Bob thus randomly permuted the key before storing it. Bob thinks that this way, even if Eve gets both the encrypted message and the key, she will not be able to read the message. Bob is not smart. Don't be like Bob.

In the above example, Bob may have, for instance, selected a permutation $(3, 4, 1, 2)$ and stored the permuted key $P = (6, 3, 16, 7)$.

One year has passed and Alice wants to decrypt her message. Only now Bob has realised that this is impossible. As he has permuted the key randomly, the message is lost forever. Did we mention that Bob isn't smart?

Bob wants to salvage at least some information from the message. Since he is not so smart, he asks for your help. You know the encrypted message $A$ and the permuted key $P$. What is the lexicographically smallest message that could have resulted in the given encrypted text?

More precisely, for given $A$ and $P$, find the lexicographically smallest message $O$, for which there exists a permutation $\pi$ such that $O_i \oplus \pi(P_i) = A_i$ for every $i$.

Note that the sequence $S$ is lexicographically smaller than the sequence $T$, if there is an index $i$ such that $S_i < T_i$ and for all $j < i$ the condition $S_j = T_j$ holds.

## Input

The first line contains a single integer $N$ ($1 \le N \le 300000$), the length of the message.

The second line contains $N$ integers $A_1, A_2, ..., A_N$ ($0 \le A_i < 2^{30}$) representing the encrypted message.

The third line contains $N$ integers $P_1, P_2, ..., P_N$ ($0 \le P_i < 2^{30}$) representing the permuted encryption key.

## Output

Output a single line with $N$ integers, the lexicographically smallest possible message $O$. Note that all its elements should be non-negative.

### Examples

| input |
| --- |
| 3<br>8 4 13<br>17 2 7 |

**output**

```
10 3 28
```

**input**

```
5
12 7 87 22 11
18 39 9 12 16
```

**output**

```
0 14 69 6 44
```

**input**

```
10
331415699 278745619 998190004 423175621 42983144 166555524 843586353 802130100 337889448
685310951
226011312 266003835 342809544 504667531 529814910 684873393 817026985 844010788 993949858
1031395667
```

**output**

```
128965467 243912600 4281110 112029883 223689619 76924724 429589 119397893 613490433
362863284
```

### Note

In the first case, the solution is $(10, 3, 28)$, since $8 \oplus 2 = 10$, $4 \oplus 7 = 3$ and $13 \oplus 17 = 28$. Other possible permutations of key yield messages $(25, 6, 10)$, $(25, 3, 15)$, $(10, 21, 10)$, $(15, 21, 15)$ and $(15, 6, 28)$, which are all lexicographically larger than the solution.