



Number Game

Robi and Aritro are good friends and math enthusiasts who frequently debate over who has the ultimate mathematical rizz and aura. To settle the debate and prove that he is the absolute goat, Aritro challenges Robi with a problem that targets his weakness in **numbers**. If Robi loses, Aritro pulls all the huzz; if Robi wins, his rizz is unmatched.

Aritro gives him an array a of n **distinct** integers greater than 1. Using normal numbers is lowkey mid for Aritro. Instead, he uses **prime powers**! Each element a_i is a prime power, meaning that $a_i = p^q$ for some prime p and integer $q \geq 1$.

Aritro challenges Robi to choose k indices, where $1 \leq k \leq n$, satisfying $i_1 < i_2 < \dots < i_k$ such that the corresponding array values form a strictly increasing subsequence. More formally, the chosen elements must satisfy:

$$a_{i_1} < a_{i_2} < \dots < a_{i_k}$$

Let $f(a_i) = q$ denote the exponent of a_i . Robi must choose a subsequence whose total exponent sum is as large as possible. More formally, output:

$$\sum_{j=1}^k f(a_{i_j})$$

Output the maximum possible value of this sum.

Input Format

- Line 1: A single integer n representing the size of the array.
- Line 2: n space-separated integers representing the elements of array a .

Output Format

- Line 1: Output a single integer representing the maximum possible sum of the exponents.

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq q \leq 59$
- $2 \leq a_i \leq 10^{18}$

- All elements in the array a are unique.

Subtasks

Subtask	Score	Additional constraints
1	5	$n = 1, a[i] \leq 10^6$
2	8	$n \leq 16, q = 1$
3	7	$n \leq 16, a[i] \leq 10^6$
4	10	$n = 1$
5	15	$n \leq 16$
6	15	$q = 1$
7	40	No additional constraints

Examples

Example 1

Input

```
4
16 27 3 2
```

Output

```
7
```

Explanation

In the first example, the prime power representations of the elements are:

- $a_1 = 16 = 2^4 \Rightarrow f(16) = 4.$
- $a_2 = 27 = 3^3 \Rightarrow f(27) = 3.$
- $a_3 = 3 = 3^1 \Rightarrow f(3) = 1.$
- $a_4 = 2 = 2^1 \Rightarrow f(2) = 1.$

The optimal strictly increasing subsequence is $[16, 27]$, obtained by choosing $i_1 = 1$ and $i_2 = 2$. The maximum possible sum of the exponents is $f(16) + f(27) = 4 + 3 = 7$.



Wifi Password

Robi is the greatest technician who ever lived. He came to visit his friend Aritro's house, where Aritro was proudly showing off his new Wi-Fi connection. Robi also wanted to test it, but asking Aritro for the Wi-Fi password would be far too ordinary for a technician of his caliber. Using his special software, Robi found the password; however, he still had to solve the puzzle given by the system to access it.

Robi has discovered two arrays of n integers, A and B . You can perform the following operation **at most once**:

- Choose an index i ($0 \leq i \leq n$) and swap $A[i]$ and $B[i]$.

For an array A , let $\text{MEX}(A)$ denote the smallest non-negative integer that does not appear in A .

For two arrays A and B of the same length, define

$$F(A, B) = \min |\text{MEX}(A) - \text{MEX}(B)|$$

where the minimum is taken over all possible ways of performing the operation above, **including not performing any swap**. In other words, $F(A, B)$ is the smallest possible gap between the two MEX values that you can achieve by after performing at most one swap.

Robi must compute, for every prefix length from 1 to n , the value F for that prefix. That is, for each k ($1 \leq k \leq n$), he needs to do the following:

$$F(A[0 \dots k-1], B[0 \dots k-1]).$$

In other words, he must compute the function for every prefix of the arrays. For each prefix, the difference between the two MEX values must be minimized.

Each prefix is evaluated **independently**; a swap performed for one prefix does not affect any other prefix.

Help Robi solve the puzzle!

Input Format

- Line 1: A single integer n representing the size of the array.
- Line 2: n space-separated integers representing the elements of array A .

- Line 3: n space-separated integers representing the elements of array B .

Output Format

- Line 1: n space-separated integers representing the answer.

Constraints

- $1 \leq n \leq 2 \times 10^5$
- $0 \leq A[i], B[i] \leq n$

Subtasks

Subtask	Score	Additional constraints
1	5	$n = 10, A[i] = B[i]$
2	15	$A[i]$ is even
3	15	$n \leq 1000$
4	20	$n \leq 5000$
5	45	No additional constraints

Examples

Example 1

Input

```
5
0 1 2 3 4
0 1 1 2 3
```

Output

```
0 0 1 1 1
```

Example 2

Input

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

Output

```
1 1 1 0
```

Explanation

In the second example, For every index i we compute like the following:

- $i = 0$: $A = [0]$, $B = [1]$. $\text{MEX}(A) = 1$ and $\text{MEX}(B) = 0$, so the answer is $|1 - 0| = 1$.
- $i = 1$: $A = [0, 4]$, $B = [1, 1]$. $\text{MEX}(A) = 1$ and $\text{MEX}(B) = 0$, so the answer is $|1 - 0| = 1$.
- $i = 2$: $A = [0, 4, 2]$, $B = [1, 1, 0]$. $\text{MEX}(A) = 1$ and $\text{MEX}(B) = 2$, so the answer is $|1 - 2| = 1$.
- $i = 3$: $A = [0, 4, 2, 2]$, $B = [1, 1, 0, 2]$. Here index 1 was swapped, so A becomes $[0, 1, 2, 2]$ and B becomes $[1, 4, 0, 2]$. Now $\text{MEX}(A) = 3$ and $\text{MEX}(B) = 3$, so the answer is $|3 - 3| = 0$.

Note

$\text{MEX}(A)$ is defined as the smallest non-negative integer that is not present in the array. For example, $\text{MEX}([1, 2, 0, 5])$ is 3, and $\text{MEX}([1, 2, 4, 9])$ is 0.



Candy Boxes

Ibor bought n boxes of candies, and each box contains exactly n candies. There is exactly one special box among them. All candies in this special box have a taste value of v , while all candies in the other $n - 1$ normal boxes have a taste value of w . It is guaranteed that the special candies taste better than the normal ones: $v > w$.

You are given the number of boxes, n . You need to determine the normal taste w , the special taste v , and the 1-based index of the special box. To do this, you can ask Ibor to perform the following procedure:

- Take any number of candies from any number of boxes. Ibor will return the total taste value of the selected candies.

You can ask Ibor at most q times.

Implementation Details

You should implement the following procedure:

```
std::vector< long long > find_candy(int n, int q)
```

- n : the number of candy boxes.
- The function must return a `std::vector` of `long long` values of size exactly 3 containing the answers in this specific order: $\{w, v, idx\}$, where idx is the 1-based index of the special box.
- q : the maximum number of times you can ask Ibor for a taste value.
- This procedure will be called exactly once by the grader.

The above procedure can make calls to the following procedure:

```
long long taste_candy(std::vector< long long > a)
```

- a : an array of size n . The element at $a[i]$ represents the number of candies you want to take from the $(i + 1)$ -th box, where $0 \leq i < n$ and $0 \leq a[i] \leq n$.
- The function returns the total taste value of the selected candies. More formally,

$$\sum_{i=0}^{n-1} a[i] \times \text{taste of the } (i+1)\text{-th box.}$$

- This procedure may be called no more than q times.

Your code must not read anything from standard input (for example, by using `scanf` or `cin`) or write anything to standard output (for example, by using `printf` or `cout`). However, you may print debugging information to standard error. For communication, use the functions described above. The grader is **not adaptive**, meaning that the values of w , v , and idx are fixed before `find_candy` is called.

Example

Let's consider the following case.

$$n = 4, \quad q = 10000, \quad w = 6, \quad v = 7, \quad idx = 3$$

The following call will be made.

```
find_candy(4, 10000)
```

This procedure may call `taste_candy` as follows.

```
taste_candy({1, 1, 1, 1})
```

Suppose the query is made using the array a :

$$a = [1, 1, 1, 1]$$

Since box 3 is special, `taste_candy(a)` returns:

$$(1 \times 6) + (1 \times 6) + (1 \times 7) + (1 \times 6) = 25.$$

Using the returned values from the queries, you must determine w , v , and idx .

Constraints

- $3 \leq n \leq 10,000$
- $1 \leq w < v \leq 1000$
- $1 \leq q \leq 10,000$

Subtasks

Subtask	Score	Additional constraints
1	40	$v = w + 1, q = 2$
2	60	No additional constraints

For the last subtask, your score is determined using a formula. Let us assume that you asked Ibor a total of X times. Your final score will be calculated using the following table:

X	Score
$X > 10^4$	0
$10 < X \leq 10^4$	$40 \times \left(1.15 - \left(\frac{\log_{10} X}{4} \right)^2 \right)$
$1 \leq X \leq 10$	80

Sample grader

Input format:

- Line 1: $n \ q \ w \ v \ idx$

Output format:

- Line 1: Returned: A
- Line 2: Expected: B
- Line 3: Queries: X

Here, A is the output of `find_candy`, B is the original solution, P is the number of all calls to `taste_candy`.

Note that the grader used for actual evaluation is different from the sample grader.