Birla Institute of Technology and Science, Pilani
Hyderabad Campus

CS F211 Data Structures and Algorithms
Lab 4: Sorting and Math

Allowed languages: **C**

General Tips

- Indent your code appropriately and use proper variable names. These increase readability and writability of the code. Also, use comments wherever necessary.
- Use a proper IDE or text editors like Sublime Text or VSCode as they help to run and test your code on multiple test-cases easily. However, in the lab you will be using command line interface.
- Try to use functions as much as possible in your code. Functions increase reusability and the pass-by-value feature provides a significant help sometimes. Modularizing your code also helps you to debug efficiently.
- Do not use any inbuilt sorting function.

Problem A. Gym Slots

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

The gymnasium of your college has been getting too crowded at times, so you have to decide on new slots for keeping the gym open.

You have a set of preferred slots, where each slot is a pair of integers representing the starting and ending time. The new slots are constructed using the following algorithm: As long as there is a pair of overlapping slots in the set, merge them to form a new slot and replace those two slots with this new slot. Keep repeating this operation until there are no two overlapping slots. The output is the set of new slots of the gym. Assume that the preferred slots given as input, will be unique.

Print the new slots that will be constructed by using the given algorithm.

Note: Two slots sl_1 and sl_2 are said to be overlapping if the ending time of sl_1 is $\geq$ the starting time of sl_2 and the starting time of sl_1 is $\leq$ the ending time of sl_2. Merging two overlapping slots produces a single new slot with start time = minimum of start time of both slots, end time = maximum of end time of both slots.
You can use arrays for this program.

The time complexity of your solution should be $O(nlogn)$, where n is the size of the array of preferred slots. Solution of higher time complexity will not be considered.

Input

The first line of input contains n ($1 \leq n \leq 10^6$) — the total number of preferred slots.

The following n lines contain two space separated integers s_i and e_i ($1 \leq s_i, e_i \leq 10^9$) — the start and end time of the i-th preferred slot, respectively.

Output

The first line of output should contain k — the number of new slots after merging all the overlapping slots.

The following k lines should contain two space separated integers $snew_i, enew_i$ — the start and end time of the i-th new slot.

Examples

standard input	standard output
4 1 3 8 10 2 6 15 18	3 1 6 8 10 15 18
2 1 4 4 5	1 1 5

Note

In Example 1, slots $[1, 3]$ and $[2, 6]$ are overlapping, so they are merged to form slot $[1, 6]$. There are no two overlapping slots after this.

In Example 2, slots $[1, 4]$ and $[4, 5]$ are overlapping, so they are merged to form slot $[1, 5]$. There are no two overlapping slots after this.

Problem B. Doggo Wars

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

It has been n days since the start of Doggo Wars at BPHC. You need to write a program to calculate the score for one dog. You will be given an array of n integers, where the i-th integer is the number of people the dog managed to bite on the i-th day. The score of the dog is x if there are x different days such that it managed to bite at least x people on each of those x days, and for the remaining $n - x$ days it bit no more than x people on each day.

If there are multiple possible values of x, the maximum value is the score.

The time complexity of your solution should be $O(nlogn)$, where n is the number of days since the start of Doggo Wars. Any solution with a higher time complexity will not be considered.

Input

The first line of input contains n ($1 \leq n \leq 10^6$) — the number of days since the start of Doggo Wars.

The second line of input contains n space separated integers $a_1, a_2, \ldots a_n$ ($1 \leq a[i] \leq 10^9$) — the number of people the dog managed to bite on each of the n days.

Output

Print one line containing the score of the dog.

Example

standard input	standard output
5 3 0 6 1 5	3

Note

In the example, on Day 1, Day 3 and Day 5 the dog managed to bite at least 3 people, and on Day 2 and Day 4 the dog bit no more than 3 people, so 3 is a possible value of x. It can be shown that no other possible value of x is more than 3, so the score of the dog is 3. Assume 1-based indexing for counting the days.

Problem C. Big Big Zeros

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

For an array a of n integers, we call a pair of integers (i, j) a *good* pair if it satisfies both of the following properties:

- $1 \leq i < j \leq n$
- The number of trailing zeros in the factorials of the i-th and the j-th elements of a are equal, i.e., $a_i!$ and $a_j!$ have the same number of trailing zeros

Given an array of integers, count the number of *good* pairs in it.

Assume 1-based indexing.

The time complexity of your solution should be $O(nlog(max(a_i)) + max(a_i))$, where n is the size of the array and $max(a_i)$ is the maximum element of the array. Any solution with a higher time complexity will not be considered.

Input

The first line of input contains n ($1 \leq n \leq 10^6$) — the size of the array.

The second line of input contains n space separated integers $a_1, a_2, \ldots a_n$ ($1 \leq a_i \leq 10^6$) — the elements of the array.

Output

Print one line containing the number of *good* pairs in the array.

Examples

standard input	standard output
6 23 20 1 3 2 26	4

Note

In the example,

- $a_1! = 23! = 25852016738884976640000$ and $a_2! = 20! = 2432902008176640000$ have 4 trailing zeros each, so $(1, 2)$ is a *good* pair.
- $a_3! = 1! = 1$ and $a_4! = 3! = 6$ do not have any trailing zeros, so $(3, 4)$ is a *good* pair.
- $a_3! = 1! = 1$ and $a_5! = 2! = 2$ do not have any trailing zeros, so $(3, 5)$ is a *good* pair.
- $a_4! = 3! = 6$ and $a_5! = 2! = 2$ do not have any trailing zeros, so $(4, 5)$ is a *good* pair.

There are no other *good* pairs in the given array.

Problem D. Count Primes

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

You are given an integer n and q queries. The i-th query consists of an integer $x_i \leq n$. For each query, print the number of primes $\leq x_i$.

Hint: Use a prime sieve.

The time complexity of your solution should be $O(nlog(log(n))+q)$. Any solution with a higher time complexity will not be considered.

Input

The first line of input contains two integers n, q ($1 \leq n, q \leq 10^6$).

The second line of input contains q space separated integers $x_1, x_2, \ldots, x_q$ ($1 \leq x_i \leq n$) — the queries.

Output

Print one line containing q space separated integers — the i-th integer should be the number of primes $\leq x_i$.

Example

standard input	standard output
157000 5 107257 57 54011 390 106350	10219 16 5502 77 10137

Problem E. Largest Coprime Divisor

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

Given two integers a and b, find the largest divisor of a that is coprime to b.

Note: Two integers are said to be coprime if the only positive integer that is a divisor of both of them is 1.

The time complexity of your solution should be $O(\sqrt{a}\ log(min\{a,b\}))$. Any solution with a higher time complexity will not be considered.

Input

The first and only line of input contains two integers a, b ($1 \leq a, b \leq 10^9$).

Output

Print one line containing the largest divisor of a that is coprime to b.

Example

standard input	standard output
30 12	5
571034878 220253782	285517439

Problem F. Bon Voyage

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

The team of Bon Voyage decides to change the way it is being organized. This is how the new system works -

1. In the first round, there are n candidates for a category and the GB vote for their preferred candidate
2. The k candidates with the highest number of votes, progress to the next round.
3. The voting is done again from these k candidates.

Each candidate is represented by a number and the votes of the first round are given as an array, the i-th element of the array denotes the candidate the i-th student voted for. Find the list of candidates that progress to the next round. (If there are multiple answers, print any)

Note - If there are two candidates with the same number of votes but only one spot is left for the next round, then you can progress any of the two to the next round.

Input

The first line consists of two integers, n and k ($1 \leq k \leq n \leq 10^6$) — the number of candidates initially and the number of candidates that will progress to the next round respectively.

The second line consists of n space separated integers $a_1, a_2, ..., a_n$ ($1 \leq a_i \leq 10^5$) — where a_i represents the candidate voted by the i-th voter.

Output

Print a single line consisting of k space separated integers — the candidates that progress to the next round (Order does not matter).

Examples

standard input	standard output
10 3 3 1 1 2 4 1 8 3 5 2	3 1 2
11 3 3 1 1 2 4 1 8 3 5 2 4	1 2 4

Note

In the first example,

- Candidate 1 has 3 votes
- Candidates 2 and 3 have 2 votes
- Candidate 4, 5 and 8 have 1 vote

So the 3 candidates with the highest number of votes are — Candidate 1, Candidate 2 and Candidate 3

In the second example,

- Candidate 1 has 3 votes
- Candidates 2, 3 and 4 have 2 votes
- Candidate 5 and 8 have 1 vote

Since we need any 3 candidates with the highest number of votes — 1 2 3, 1 2 4, 1 3 4 are all valid answers

Problem G. Radix Sort

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

The *radix sort* algorithm sorts an array of integers by sorting it digit by digit starting from the unit's place to the most significant digit. In each "*pass*" of the algorithm, the array is sorted based on a digit —

1. In the first pass, the array is sorted by the unit's place digit
2. In the second pass, the array is sorted by the tenth's place digit

And so on

Given an array, sort the array using the *radix sort* algorithm and print the array after each pass.

Note — Use *counting sort* while sorting based on the digits

Input

The first line of the input consists of a single integer n ($1 \leq n \leq 10^6$) — the number of elements in the array.

The second line contains n space separated positive integers $a_1, a_2, ..., a_n$ ($1 \leq a_i \leq 10^9$) — the elements of the array.

Output

Print k lines where k is the number of *passes* in the algorithm and in the i-th line print the elements of the array after the i-th pass.

Examples

standard input	standard output
6 37 156 89 41 4 19	41 4 156 37 89 19 4 19 37 41 156 89 4 19 37 41 89 156

Problem H. Allies

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

The king of Atlantis, Atlan holds a grand dinner for kings from across the planet. n kings attend the dinner and each king is assigned to one of the m banquet halls.

Each banquet hall needs to host at least one king (i.e., no banquet hall can go unassigned and a banquet hall can be assigned to more than one king). After the dinner, all pairs of kings in the same banquet hall become key allies.

Note that only a pair of kings can form an alliance.

Atlan wants to know the minimum and the maximum number of pairs of allies that can be formed after the dinner and wants your help in finding it.

Input

The first and only line consists of two space separated integers, n and m ($1 \leq m \leq n \leq 10^5$) — the number of kings attending the dinner and the number of banquet halls available respectively.

Use `long long` to prevent errors due to integer overflow

Output

Print two space separated integers denoting the minimum and the maximum number of pairs of allies that can be formed after the dinner.

Examples

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

Note

In the first example, there is only one 1 banquet hall. So, all 4 kings will be assigned to that. The number of pairs of allies will be $\binom{4}{2} = 6$

In the second example,
For minimum - one banquet hall will host 2 kings and the other will host 3 kings forming 4 pairs of allies.
For maximum - one banquet hall will host 1 king and the other will host 4 kings forming 6 pairs of allies.

Problem I. Bad Number

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

You are given an array of n positive integers, your task is to remove an element from the array such that the *gcd* of the rest of the elements is maximized. Find the maximum possible *gcd* of the array that can be obtained after removing an element.

Input

The first line of the input consists of a single integer n ($1 \leq n \leq 10^6$) — the number of elements in the array.

The second line contains n space separated positive integers $a_1, a_2, ..., a_n$ ($1 \leq a_i \leq 10^6$) — the elements of the array.

Output

Print a single integer — the maximum *gcd* of the array after removing an element from it

Examples

standard input	standard output
6 2 3 6 4 10 2	2
5 4 2 1 8 3	1

Note

In the first example, we can remove 3 and the *gcd* of the rest of the elements is 2 which is the maximum possible.

In the second example, no matter what element we remove, the *gcd* will still be 1.

Problem J. Equal Chocolates

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

It's your birthday and you brought chocolates to class. There are n students in your class (excluding you) and each student takes a_i chocolates. Some students are greedy and end up taking more chocolates than the others. You decide to perform some operations such that the number of chocolates each student has is the same.

*In each operation, you can **either** take a chocolate from a student **or** give a student an additional chocolate.*

Since you are tired, you want to perform the minimum number of operations possible.

Input

The first line contains two integers, n ($1 \leq n \leq 10^6$) — the number of students.

The second line contains n space separated positive integers $a_1, a_2, ..., a_n$ ($1 \leq a_i \leq 10^9$) — where a_i is the number of chocolates taken by the i^{th} student.

Use `long long` to prevent errors due to integer overflow.

Output

A single integer denoting the number of operations you performed such that every student has the same number of chocolates.

Example

standard input	standard output
5 1 5 2 3 7	9

Note

Assume 1-based indexing for the explanation.

1. We will take one chocolate from student 5 — s/he now has 6
2. We will take one chocolate from student 5 — s/he now has 5
3. We will take one chocolate from student 5 — s/he now has 4
4. We will take one chocolate from student 5 — s/he now has 3
5. We will take one chocolate from student 2 — s/he now has 4
6. We will take one chocolate from student 2 — s/he now has 3
7. We will add one chocolate to student 1 — s/he now has 2
8. We will add one chocolate to student 1 — s/he now has 3
9. We will add one chocolate to student 3 — s/he now has 3

After 9 operations, all students will have 3 chocolates