Waiter

HackerRank

You are a waiter at a party. There is a pile of numbered plates. Create an empty $answers$ array. At each iteration, i, remove each plate from the top of the stack in order. Determine if the number on the plate is evenly divisible by the i^th prime number. If it is, stack it in pile B_i. Otherwise, stack it in stack A_i. Store the values in B_i from top to bottom in $answers$. In the next iteration, do the same with the values in stack A_i. Once the required number of iterations is complete, store the remaining values in A_i in $answers$, again from top to bottom. Return the $answers$ array.

Example

$A = [2, 3, 4, 5, 6, 7]$
$q = 3$

An abbreviated list of primes is $[2, 3, 5, 7, 11, 13]$. Stack the plates in reverse order.

$A_0 = [2, 3, 4, 5, 6, 7]$
$answers = []$

Begin iterations. On the first iteration, check if items are divisible by 2.
$A_1 = [7, 5, 3]$
$B_1 = [6, 4, 2]$

Move B_1 elements to $answers$.

$answers = [2, 4, 6]$

On the second iteration, test if A_1 elements are divisible by 3.
$A_2 = [7, 5]$
$B_2 = [3]$

Move B_2 elmements to $answers$.

$answers = [2, 4, 6, 3]$

And on the third iteration, test if A_2 elements are divisible by 5.
$A_3 = [7]$
$B_3 = [5]$

Move B_2 elmements to $answers$.

$answers = [2, 4, 6, 3, 5]$

All iterations are complete, so move the remaining elements in A_3, from top to bottom, to $answers$.

$answers = [2, 4, 6, 3, 5, 7]$. Return this list.

Function Description

Complete the *waiter* function in the editor below.

waiter has the following parameters:

- *int number[n]:* the numbers on the plates
- *int q:* the number of iterations

Returns

- *int[n]:* the numbers on the plates after processing

Input Format

The first line contains two space separated integers, n and q.
The next line contains n space separated integers representing the initial pile of plates, i.e., A.

Constraints

$1 \leq n \leq 5 \times 10^4$
$2 \leq number[i] \leq 10^4$
$1 \leq q \leq 1200$

Sample Input

```
5 1
3 4 7 6 5
```

Sample Output

```
4
6
3
7
5
```

Explanation

Initially:

A_0 = [3, 4, 7, 6, 5]<-TOP

After 1 iteration:

A_0 = []<-TOP

B_1 = [6, 4]<-TOP

A_1 = [5, 7, 3]<-TOP

We should output numbers in B_1 first from top to bottom, and then output numbers in A_1 from top to bottom.