Codeforces Round #708 (Div. 2)

A. Meximization

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

You are given an integer n and an array $a_1, a_2, \ldots, a_n$. You should reorder the elements of the array a in such way that the sum of $\textbf{MEX}$ on prefixes (i-th prefix is $a_1, a_2, \ldots, a_i$) is maximized.

Formally, you should find an array $b_1, b_2, \ldots, b_n$, such that the sets of elements of arrays a and b are equal (it is equivalent to array b can be found as an array a with some reordering of its elements) and $\sum_{i=1}^{n} \textbf{MEX}(b_1, b_2, \ldots, b_i)$ is maximized.

$\textbf{MEX}$ of a set of nonnegative integers is the minimal nonnegative integer such that it is not in the set.

For example, $\textbf{MEX}(\{1, 2, 3\}) = 0$, $\textbf{MEX}(\{0, 1, 2, 4, 5\}) = 3$.

Input

The first line contains a single integer t $(1 \le t \le 100)$ — the number of test cases.

The first line of each test case contains a single integer n $(1 \le n \le 100)$.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i \le 100)$.

Output

For each test case print an array $b_1, b_2, \ldots, b_n$ — the optimal reordering of $a_1, a_2, \ldots, a_n$, so the sum of $\textbf{MEX}$ on its prefixes is maximized.

If there exist multiple optimal answers you can find any.

Example

input
3 7 4 2 0 1 3 3 7 5 2 2 8 6 9 1 0
output
0 1 2 3 4 7 3 2 6 8 9 2 0

Note

In the first test case in the answer $\textbf{MEX}$ for prefixes will be:

1. $\textbf{MEX}(\{0\}) = 1$
2. $\textbf{MEX}(\{0, 1\}) = 2$
3. $\textbf{MEX}(\{0, 1, 2\}) = 3$
4. $\textbf{MEX}(\{0, 1, 2, 3\}) = 4$
5. $\textbf{MEX}(\{0, 1, 2, 3, 4\}) = 5$
6. $\textbf{MEX}(\{0, 1, 2, 3, 4, 7\}) = 5$
7. $\textbf{MEX}(\{0, 1, 2, 3, 4, 7, 3\}) = 5$

The sum of $\textbf{MEX} = 1 + 2 + 3 + 4 + 5 + 5 + 5 = 25$. It can be proven, that it is a maximum possible sum of $\textbf{MEX}$ on prefixes.

B. M-arrays

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

You are given an array $a_1, a_2, \ldots, a_n$ consisting of n positive integers and a positive integer m.

You should divide elements of this array into some arrays. You can order the elements in the new arrays as you want.

Let's call an array m-divisible if for each two adjacent numbers in the array (two numbers on the positions i and $i+1$ are called adjacent for each i) their sum is divisible by m. An array of one element is m-divisible.

Find the smallest number of m-divisible arrays that $a_1, a_2, \ldots, a_n$ is possible to divide into.

Input
The first line contains a single integer t $(1 \le t \le 1000)$ — the number of test cases.

The first line of each test case contains two integers n, m $(1 \le n \le 10^5, 1 \le m \le 10^5)$.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^9)$.

It is guaranteed that the sum of n and the sum of m over all test cases do not exceed 10^5.

Output
For each test case print the answer to the problem.

Example

input
4 6 4 2 2 8 6 9 4 10 8 1 1 1 5 2 4 4 8 6 7 1 1 666 2 2 2 4
output
3 6 1 1

Note
In the first test case we can divide the elements as follows:

- $[4, 8]$. It is a 4-divisible array because $4+8$ is divisible by 4.
- $[2, 6, 2]$. It is a 4-divisible array because $2+6$ and $6+2$ are divisible by 4.
- $[9]$. It is a 4-divisible array because it consists of one element.

C1. k-LCM (easy version)

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

It is the easy version of the problem. The only difference is that in this version $k = 3$.

You are given a positive integer n. Find k positive integers $a_1, a_2, \ldots, a_k$, such that:

- $a_1 + a_2 + \ldots + a_k = n$
- $LCM(a_1, a_2, \ldots, a_k) \le \frac{n}{2}$

Here LCM is the least common multiple of numbers $a_1, a_2, \ldots, a_k$.

We can show that for given constraints the answer always exists.

Input
The first line contains a single integer t $(1 \le t \le 10^4)$ — the number of test cases.

The only line of each test case contains two integers n, k $(3 \le n \le 10^9, k = 3)$.

Output
For each test case print k positive integers $a_1, a_2, \ldots, a_k$, for which all conditions are satisfied.

Example

input
3 3 3 8 3 14 3
output
1 1 1 4 2 2 2 6 6

C2. k-LCM (hard version)

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

It is the hard version of the problem. The only difference is that in this version $3 \le k \le n$.

You are given a positive integer n. Find k positive integers $a_1, a_2, \ldots, a_k$, such that:

- $a_1 + a_2 + \ldots + a_k = n$
- $LCM(a_1, a_2, \ldots, a_k) \le \frac{n}{2}$

Here LCM is the least common multiple of numbers $a_1, a_2, \ldots, a_k$.

We can show that for given constraints the answer always exists.

Input

The first line contains a single integer t $(1 \le t \le 10^4)$ — the number of test cases.

The only line of each test case contains two integers n, k $(3 \le n \le 10^9, 3 \le k \le n)$.

It is guaranteed that the sum of k over all test cases does not exceed 10^5.

Output

For each test case print k positive integers $a_1, a_2, \ldots, a_k$, for which all conditions are satisfied.

Example

input
2 6 4 9 5
output
1 2 2 1 1 3 3 1 1

D. Genius

time limit per test: 2 seconds
memory limit per test: 32 megabytes
input: standard input
output: standard output

Please note the non-standard memory limit.

There are n problems numbered with integers from 1 to n. i-th problem has the complexity $c_i = 2^i$, tag tag_i and score s_i.

After solving the problem i it's allowed to solve problem j if and only if $\text{IQ} < |c_i - c_j|$ and $tag_i \ne tag_j$. After solving it your IQ changes and becomes $\text{IQ} = |c_i - c_j|$ and you gain $|s_i - s_j|$ points.

Any problem can be the first. You can solve problems in any order and as many times as you want.

Initially your $\text{IQ} = 0$. Find the maximum number of points that can be earned.

Input

The first line contains a single integer t $(1 \le t \le 100)$ — the number of test cases.

The first line of each test case contains an integer n $(1 \le n \le 5000)$ — the number of problems.

The second line of each test case contains n integers $tag_1, tag_2, \ldots, tag_n$ $(1 \le tag_i \le n)$ — tags of the problems.

The third line of each test case contains n integers $s_1, s_2, \ldots, s_n$ $(1 \le s_i \le 10^9)$ — scores of the problems.

It's guaranteed that sum of n over all test cases does not exceed 5000.

Output

For each test case print a single integer — the maximum number of points that can be earned.

Example

input
5 4 1 2 3 4 5 10 15 20 4 1 2 1 2

```
5 10 15 20
4
2 2 4 1
2 8 19 1
2
1 1
6 9
1
1
666
```

output

```
35
30
42
0
0
```

Note

In the first test case optimal sequence of solving problems is as follows:

1. $1 \to 2$, after that total score is 5 and $\mathrm{IQ} = 2$
2. $2 \to 3$, after that total score is 10 and $\mathrm{IQ} = 4$
3. $3 \to 1$, after that total score is 20 and $\mathrm{IQ} = 6$
4. $1 \to 4$, after that total score is 35 and $\mathrm{IQ} = 14$

In the second test case optimal sequence of solving problems is as follows:

1. $1 \to 2$, after that total score is 5 and $\mathrm{IQ} = 2$
2. $2 \to 3$, after that total score is 10 and $\mathrm{IQ} = 4$
3. $3 \to 4$, after that total score is 15 and $\mathrm{IQ} = 8$
4. $4 \to 1$, after that total score is 35 and $\mathrm{IQ} = 14$

In the third test case optimal sequence of solving problems is as follows:

1. $1 \to 3$, after that total score is 17 and $\mathrm{IQ} = 6$
2. $3 \to 4$, after that total score is 35 and $\mathrm{IQ} = 8$
3. $4 \to 2$, after that total score is 42 and $\mathrm{IQ} = 12$

E1. Square-free division (easy version)

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

This is the easy version of the problem. The only difference is that in this version $k = 0$.

There is an array $a_1, a_2, \ldots, a_n$ of n positive integers. You should divide it into a minimal number of continuous segments, such that in each segment there are no two numbers (on different positions), whose product is a perfect square.

Moreover, it is allowed to do at most k such operations before the division: choose a number in the array and change its value to any positive integer. But in this version $k = 0$, so it is not important.

What is the minimum number of continuous segments you should use if you will make changes optimally?

Input

The first line contains a single integer t $(1 \le t \le 1000)$ — the number of test cases.

The first line of each test case contains two integers n, k ($1 \le n \le 2 \cdot 10^5$, $k = 0$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^7$).

It's guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case print a single integer — the answer to the problem.

Example

input

```
3
5 0
18 6 2 4 1
5 0
6 8 1 24 8
1 0
1
```

output

```
3
```

```
2
1
```

Note
In the first test case the division may be as follows:

- $[18, 6]$
- $[2, 4]$
- $[1]$

E2. Square-free division (hard version)

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

This is the hard version of the problem. The only difference is that in this version $0 \le k \le 20$.

There is an array $a_1, a_2, \ldots, a_n$ of n positive integers. You should divide it into a minimal number of continuous segments, such that in each segment there are no two numbers (on different positions), whose product is a perfect square.

Moreover, it is allowed to do at most k such operations before the division: choose a number in the array and change its value to any positive integer.

What is the minimum number of continuous segments you should use if you will make changes optimally?

Input
The first line contains a single integer t $(1 \le t \le 1000)$ — the number of test cases.

The first line of each test case contains two integers n, k $(1 \le n \le 2 \cdot 10^5, 0 \le k \le 20)$.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^7)$.

It's guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output
For each test case print a single integer — the answer to the problem.

Example

input
3 5 2 18 6 2 4 1 11 4 6 2 2 8 9 1 3 6 3 9 7 1 0 1
output
1 2 1

Note
In the first test case it is possible to change the array this way: $[\underline{3}, 6, 2, 4, \underline{5}]$ (changed elements are underlined). After that the array does not need to be divided, so the answer is 1.

In the second test case it is possible to change the array this way: $[6, 2, \underline{3}, 8, 9, \underline{5}, 3, 6, \underline{10}, \underline{11}, 7]$. After that such division is optimal:

- $[6, 2, 3]$
- $[8, 9, 5, 3, 6, 10, 11, 7]$