Educational Codeforces Round 131 (Rated for Div. 2)

A. Grass Field

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

There is a field of size 2×2. Each cell of this field can either contain grass or be empty. The value $a_{i,j}$ is 1 if the cell (i, j) contains grass, or 0 otherwise.

In one move, you can choose **one row** and **one column** and cut all the grass in this row and this column. In other words, you choose the row x and the column y, then you cut the grass in all cells $a_{x,i}$ and all cells $a_{i,y}$ for all i from 1 to 2. After you cut the grass from a cell, it becomes empty (i. e. its value is replaced by 0).

Your task is to find the minimum number of moves required to cut the grass in all non-empty cells of the field (i. e. make all $a_{i,j}$ zeros).

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 16$) — the number of test cases. Then t test cases follow.

The test case consists of two lines, each of these lines contains two integers. The j-th integer in the i-th row is $a_{i,j}$. If $a_{i,j} = 0$ then the cell (i, j) is empty, and if $a_{i,j} = 1$ the cell (i, j) contains grass.

Output

For each test case, print one integer — the minimum number of moves required to cut the grass in all non-empty cells of the field (i. e. make all $a_{i,j}$ zeros) in the corresponding test case.

Example

input
3 0 0 0 0 1 0 0 1 1 1 1 1
output
0 1 2

B. Permutation

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

Recall that a permutation of length n is an array where each element from 1 to n occurs exactly once.

For a fixed positive integer d, let's define the cost of the permutation p of length n as the number of indices i $(1 \le i < n)$ such that $p_i \cdot d = p_{i+1}$.

For example, if $d = 3$ and $p = [5, 2, 6, 7, 1, 3, 4]$, then the cost of such a permutation is 2, because $p_2 \cdot 3 = p_3$ and $p_5 \cdot 3 = p_6$.

Your task is the following one: for a given value n, find the permutation of length n and the value d with maximum possible cost (over all ways to choose the permutation and d). If there are multiple answers, then print any of them.

Input

The first line contains a single integer t ($1 \le t \le 500$) — the number of test cases.

The single line of each test case contains a single integer n ($2 \le n \le 2 \cdot 10^5$).

The sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print the value d in the first line, and n integers in the second line — the permutation itself. If there are multiple answers, then print any of them.

Example

input
2 2 3
output
2 1 2 3 2 1 3

C. Schedule Management

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

There are n workers and m tasks. The workers are numbered from 1 to n. Each task i has a value a_i — the index of worker who is proficient in this task.

Every task should have a worker assigned to it. If a worker is proficient in the task, they complete it in 1 hour. Otherwise, it takes them 2 hours.

The workers work in parallel, independently of each other. Each worker can only work on one task at once.

Assign the workers to all tasks in such a way that the tasks are completed as early as possible. The work starts at time 0. What's the minimum time all tasks can be completed by?

Input

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

The first line of each testcase contains two integers n and m ($1 \le n \le m \le 2 \cdot 10^5$) — the number of workers and the number of tasks.

The second line contains m integers $a_1, a_2, \ldots, a_m$ ($1 \le a_i \le n$) — the index of the worker proficient in the i-th task.

The sum of m over all testcases doesn't exceed $2 \cdot 10^5$.

Output

For each testcase, print a single integer — the minimum time all tasks can be completed by.

Example

input
4 2 4 1 2 1 2 2 4 1 1 1 1 5 5 5 1 3 2 4 1 1 1
output
2 3 1 1

Note

In the first testcase, the first worker works on tasks 1 and 3, and the second worker works on tasks 2 and 4. Since they both are proficient in the corresponding tasks, they take 1 hour on each. Both of them complete 2 tasks in 2 hours. Thus, all tasks are completed by 2 hours.

In the second testcase, it's optimal to assign the first worker to tasks $1, 2$ and 3 and the second worker to task 4. The first worker spends 3 hours, the second worker spends 2 hours (since they are not proficient in the taken task).

In the third example, each worker can be assigned to the task they are proficient at. Thus, each of them complete their task in 1 hour.

D. Permutation Restoration

time limit per test: 4 seconds
memory limit per test: 256 megabytes

input: standard input
output: standard output

Monocarp had a permutation a of n integers 1, 2, ..., n (a permutation is an array where each element from 1 to n occurs exactly once).

Then Monocarp calculated an array of integers b of size n, where $b_i = \left\lfloor \frac{i}{a_i} \right\rfloor$. For example, if the permutation a is $[2, 1, 4, 3]$, then the array b is equal to $\left[\left\lfloor \frac{1}{2} \right\rfloor, \left\lfloor \frac{2}{1} \right\rfloor, \left\lfloor \frac{3}{4} \right\rfloor, \left\lfloor \frac{4}{3} \right\rfloor\right] = [0, 2, 0, 1]$.

Unfortunately, the Monocarp has lost his permutation, so he wants to restore it. Your task is to find a permutation a that corresponds to the given array b. If there are multiple possible permutations, then print any of them. The tests are constructed in such a way that least one suitable permutation exists.

Input
The first line contains a single integer t ($1 \le t \le 10^5$) — number of test cases.

The first line of each test case contains a single integer n ($1 \le n \le 5 \cdot 10^5$).

The second line contains n integers $b_1, b_2, \ldots, b_n$ ($0 \le b_i \le n$).

Additional constrains on the input:

- the sum of n over test cases does not exceed $5 \cdot 10^5$;
- there exists at least one permutation a that would yield this array b.

Output
For each test case, print n integers — a permutation a that corresponds to the given array b. If there are multiple possible permutations, then print any of them.

Example

input
4 4 0 2 0 1 2 1 1 5 0 0 1 4 1 3 0 1 3
output
2 1 4 3 1 2 3 4 2 1 5 3 2 1

E. Text Editor

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

You wanted to write a text t consisting of m lowercase Latin letters. But instead, you have written a text s consisting of n lowercase Latin letters, and now you want to fix it by obtaining the text t from the text s.

Initially, the cursor of your text editor is at the end of the text s (after its last character). In one move, you can do one of the following actions:

- press the "left" button, so the cursor is moved to the left by one position (or does nothing if it is pointing at the beginning of the text, i. e. before its first character);
- press the "right" button, so the cursor is moved to the right by one position (or does nothing if it is pointing at the end of the text, i. e. after its last character);
- press the "home" button, so the cursor is moved to the beginning of the text (before the first character of the text);
- press the "end" button, so the cursor is moved to the end of the text (after the last character of the text);
- press the "backspace" button, so the character before the cursor is removed from the text (if there is no such character, nothing happens).

Your task is to calculate the minimum number of moves required to obtain the text t from the text s using the given set of actions, or determine it is impossible to obtain the text t from the text s.

You have to answer T independent test cases.

Input
The first line of the input contains one integer T ($1 \le T \le 5000$) — the number of test cases. Then T test cases follow.

The first line of the test case contains two integers n and m ($1 \le m \le n \le 5000$) — the length of s and the length of t, respectively.

The second line of the test case contains the string s consisting of n lowercase Latin letters.

The third line of the test case contains the string t consisting of m lowercase Latin letters.

It is guaranteed that the sum of n over all test cases does not exceed 5000 ($\sum n \le 5000$).

Output

For each test case, print one integer — the minimum number of moves required to obtain the text t from the text s using the given set of actions, or `-1` if it is impossible to obtain the text t from the text s in the given test case.

Example

input
6 9 4 aaaaaaaaa aaaa 7 3 abacaba aaa 5 4 aabcd abcd 4 2 abba bb 6 4 baraka baka 8 7 question problem
output
5 6 3 4 4 -1

F. Points

time limit per test: 6.5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

A triple of points i, j and k on a coordinate line is called **beautiful** if $i < j < k$ and $k - i \le d$.

You are given a set of points on a coordinate line, initially empty. You have to process queries of three types:

- add a point;
- remove a point;
- calculate the number of beautiful triples consisting of points belonging to the set.

Input

The first line contains two integers q and d ($1 \le q, d \le 2 \cdot 10^5$) — the number of queries and the parameter for defining if a triple is beautiful, respectively.

The second line contains q integers $a_1, a_2, \ldots, a_q$ ($1 \le a_i \le 2 \cdot 10^5$) denoting the queries. The integer a_i denotes the i-th query in the following way:

- if the point a_i belongs to the set, remove it; otherwise, add it;
- after adding or removing the point, print the number of beautiful triples.

Output

For each query, print one integer — the number of beautiful triples after processing the respective query.

Example

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

```
5
1
5
```