Codeforces Global Round 29 (Div. 1 + Div. 2)

A. Shortest Increasing Path

1 second, 256 megabytes

You are at $(0,0)$ in a rectangular grid and want to go to (x,y).

In order to do so, you are allowed to perform a sequence of steps.

Each step consists of moving a positive integer amount of length in the positive direction of either the x or the y axis.

The first step must be along the x axis, the second along the y axis, the third along the x axis, and so on. Formally, if we number steps from one in the order they are done, then odd-numbered steps must be along the x axis and even-numbered steps must be along the y axis.

Additionally, each step must have a length **strictly greater** than the length of the previous one.

Output the minimum number of steps needed to reach (x,y), or -1 if it is impossible.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first and only line of each case contains two integers x and y ($1 \le x,y \le 10^9$).

Output

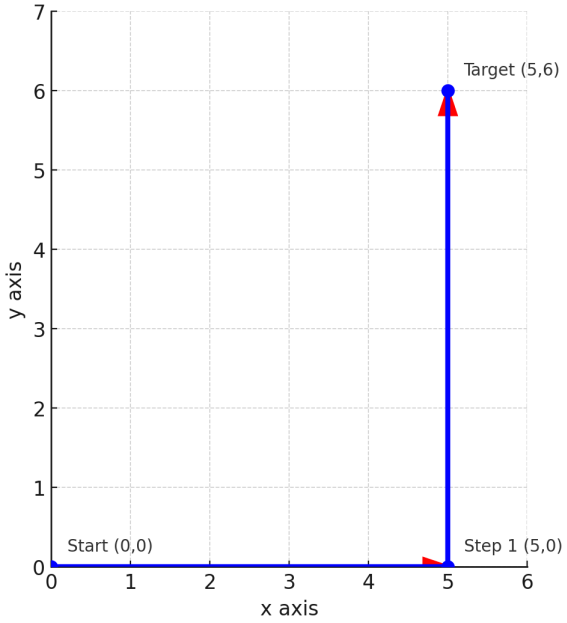
For each test case, output the minimum number of steps to reach (x,y) or -1 if it is impossible.

input
10 1 2 5 6 4 2 1 1 2 1 3 3 5 1 5 4 752 18572 95152 2322

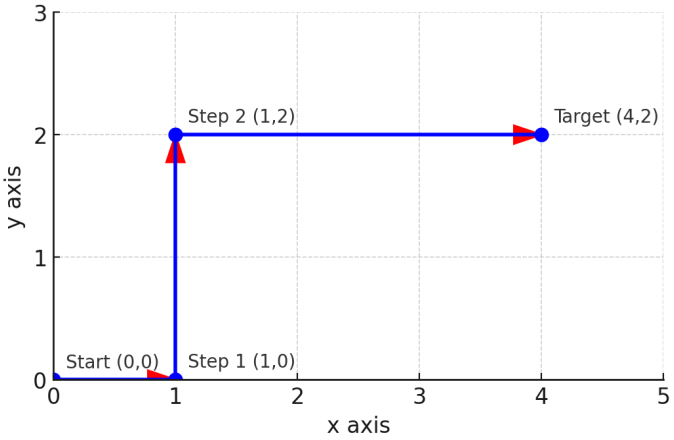
output
2 2 3 -1 -1 -1 -1 2 3

Visualizer link

In the second test case, you can move to $(5,0)$ by moving 5 along the x axis and then to $(5,6)$ by moving 6 along the y axis.



In the third test case, you can move to $(1,0)$, then to $(1,2)$, and finally to $(4,2)$.



In the fourth test case, reaching $(1,1)$ is impossible since after moving to $(1,0)$ along the x axis, you are forced to move at least 2 along the y axis.

B. Multiple Construction

1 second, 256 megabytes

You are given an integer n. Your task is to construct an array of length $2 \cdot n$ such that:

- Each integer from 1 to n appears exactly twice in the array.
- For each integer x ($1 \le x \le n$), the distance between the two occurrences of x is a multiple of x. In other words, if p_x and q_x are the indices of the two occurrences of x, $|q_x - p_x|$ must be divisible by x.

It can be shown that a solution always exists.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

Each of the next t lines contains a single integer n ($1 \le n \le 2 \cdot 10^5$).

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

Output

For each test case, print a line containing $2 \cdot n$ integers — the array that satisfies the given conditions.

If there are multiple valid answers, print any of them.

input
3 2 3 1

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

Visualizer link

In the first test case:

- The number 1 appears at positions 1 and 3: the distance is 2, which is divisible by 1.
- The number 2 appears at positions 2 and 4: the distance is 2, which is divisible by 2.

In the second test case:

- The number 1 appears at positions 1 and 3: the distance is 2, which is divisible by 1.
- The number 2 appears at positions 4 and 6: the distance is 2, which is divisible by 2.
- The number 3 appears at positions 2 and 5: the distance is 3, which is divisible by 3.

In the third test case, the two occurrences of 1 are at positions 1 and 2, so the distance between them is 1, which is a multiple of 1.

C. Rabbits

2 seconds, 256 megabytes

You have n flower pots arranged in a line numbered from 1 to n left to right. Some of the pots contain flowers, while others are empty. You are given a binary string s describing which pots contain flowers ($s_i = 1$) and which are empty ($s_i = 0$). You also have some rabbits, and you want to take a nice picture of rabbits and flowers. You want to put rabbits in every empty pot ($s_i = 0$), and for each rabbit, you can put it looking either to the left or to the right. Unfortunately, the rabbits are quite naughty, and they will try to jump, which will ruin the picture.

Each rabbit will prepare to jump into the next pot in the direction they are looking, but they won't jump if there is a rabbit in that pot already or if there is another rabbit that prepares to jump into the same pot from the opposite side. Rabbits won't jump out of the borders (a rabbit at pot 1 looking to the left won't jump, same for a rabbit looking to the right at pot n).

Your goal is to choose the directions of the rabbits so that they never jump, allowing you to take your time to take the picture. You need to determine if there is a valid arrangement of rabbits such that no rabbit ever jumps.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains an integer n ($1 \le n \le 2 \cdot 10^5$).

The second line contains a binary string s of size n, denoting the occupied and empty pots.

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

Output

For each test case, print "YES" if there exists a configuration of rabbits that satisfies the condition, and "NO" otherwise.

You can output the answer in any case (upper or lower). For example, the strings "yEs", "yes", "Yes", and "YES" will be recognized as positive responses.

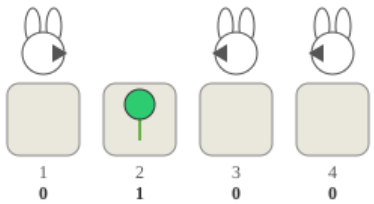
input
12 4 0100 3 000 8 11011011 5 00100 1 1 5 01011 2 01 7 0101011 7 1101010 5 11001 4 1101 9 001101100

output
YES YES NO YES YES YES YES YES YES NO NO

Visualizer link

In the first test case, one of the valid configurations is to put a rabbit looking to the right at position 1, a rabbit looking to the left at position 3, and a rabbit looking to the left at position 4. No rabbit will move since:

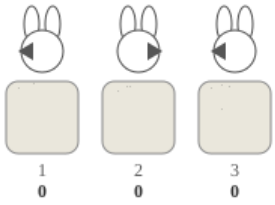
- The rabbit at pot 1 won't move to pot 2 since the rabbit at pot 3 is looking to the left.
- The rabbit at pot 3 won't move to pot 2 since the rabbit at pot 1 is looking to the right.
- The rabbit at pot 4 won't move to pot 3 since there is a rabbit in there.



In the second test case, one of the valid configurations is to put a rabbit looking to the left at position 1, a rabbit looking to the right at position 2, and a rabbit looking to the left at position 3. No rabbit will move since:

- The rabbit at pot 1 won't move since it is looking at the left border.

- The rabbit at pot 2 won't move to pot 3 since there is a rabbit in there.
- The rabbit at pot 3 won't move to pot 2 since there is a rabbit in there.



It can be proven that there is no valid arrangement of rabbits in the third test case.

D. Game on Array

2 seconds, 256 megabytes

You are given an array a of n positive integers. Alice and Bob will play a game with this array. They will take alternating turns, with Alice going first.

At each turn, the player must choose a value $x > 0$ that appears in a at least once. Then,

1. the player earns 1 point for each value x in the array,
2. each value x in the array is decreased by 1 and becomes $x - 1$.

Note that the player can choose x only if it is present in a at the moment, so each valid move earns a positive amount of points. For example, if the array is $[3, 8, 5, 8]$ and Alice chooses $x = 8$, the array will become $[3, 7, 5, 7]$ and Alice will earn 2 points.
The game ends when no x can be chosen; that is, when all the elements in the array are zero.

Given that both players want to maximize their points and play optimally, calculate the amount of points that each player will end up with.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^3$). The description of the test cases follows.

The first line of each test case contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the size of the array.

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

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

Output

For each test case, output two integers, the amount of points Alice and Bob will get if both play optimally.

input
3 3 2 1 1 5 3 3 3 5 5 4 9 9 9 9

output
3 1 10 9 20 16

Visualizer link

In the first test case, Alice chooses $x = 1$ with her first move. The array becomes $[2, 0, 0]$ and she earns 2 points. After that, Bob is forced to choose $x = 2$. The array becomes $[1, 0, 0]$ and Bob earns 1 point. Finally, Alice is forced to choose $x = 1$ and earn one more point. After that, the array is $[0, 0, 0]$, so the game ends. In total, Alice finishes with 3 points and Bob with 1 point.

In the third test case, each player will decrement all the elements and earn 4 points each move. Alice will earn $5 \cdot 4 = 20$ points while Bob will only earn $4 \cdot 4 = 16$ points.

E. Maximum OR Popcount

2 seconds, 256 megabytes

You are given an array of n non-negative integers.

You want to answer given q independent scenarios. In the i-th scenario, you are allowed to perform the following operation **at most** b_i times:

- Choose an element of the array and increase it by 1.

Your goal is to maximize the number of bits that are equal to 1 in the bitwise OR of all numbers in the array. Find this number for each scenario.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^3$). The description of the test cases follows.

The first line of each test case contains two integers n and q ($1 \le n, q \le 10^5$) — the size of the array and the number of scenarios.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$) — the elements of the array.

The i-th of the next q lines contains a single integer b_i ($0 \le b_i \le 10^9$) — the maximum number of operations allowed in the i-th scenario.

It is guaranteed that the sum of n over all test cases does not exceed 10^5, and the sum of q over all test cases does not exceed 10^5.

Output

For each test case, output q lines, the i-th of them containing a single integer — the maximum possible number of bits equal to 1 in the bitwise OR in the i-th scenario.

input
3 1 3 0 0 2 4 2 2 1 3 0 3 2 1 1000000000 1000000000 1000000000

output
0 1 2 2 3 31

Visualizer link

In the first test case:

- In the first scenario, we don't have any operations, and therefore the answer is equal to the number of 1-bits in the bitwise OR of the original array, 0, which has 0 bits set in its binary representation.
- In the second scenario, one way of achieving 1 bit set in the bitwise OR is increasing a_1 by 1 twice, obtaining a bitwise OR of $2 = (10)_2$.

It can be shown that it is the best possible value we can obtain by performing the operation at most twice.

- In the third scenario, one way of achieving a result of 2 is adding 1 to a_1 three times, which can be shown to be optimal. Note that you don't have to apply the operation 4 times.

In the second test case:

- In the first scenario, we don't have any operations, and therefore the answer is equal to the number of 1-bits in the bitwise OR of the original array, which is 2.
- In the second scenario, one way of achieving a result of 3 is adding 1 to a_2 three times, which can be shown to be optimal.

F. Exchange Queries

2 seconds, 256 megabytes

You are at a marketplace and there are two traders who are willing to trade on n different items. Each trader is represented by a permutation that signifies the relative value of the n items for that trader. Let's denote those two permutations p and s. If you have an item i, you can trade it for an item j if either $p_i > p_j$ (using the first trader) or $s_i > s_j$ (using the second trader).

We say that an item i is *at least as valuable as* an item j if you can come to the marketplace with item i, make some (possibly none) trades, and get item j. Note that at any moment, you will have exactly one item on hand. Assume that the traders have infinite supplies of any item.

Due to the never-ending updates in the market, traders will often reevaluate their views on the item values. You will be given q queries; each is a swap in one of the permutations. After every update, you should print the number of pairs (i, j) $(1 \le i, j \le n)$ such that item i is at least as valuable as item j.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains two integers n and q ($2 \le n \le 10^5$, $1 \le q \le 10^5$).

The next two lines contain the initial permutations p and s.

The following q lines each contain three integers tp, i, j ($tp \in \{1, 2\}$, $1 \le i, j \le n$, $i \ne j$). If $tp = 1$, swap p_i with p_j. If $tp = 2$, swap s_i with s_j.

It is guaranteed that the sum of n over all test cases does not exceed 10^5, and the sum of q over all test cases does not exceed 10^5.

Output

For each test case, output q lines, the k-th of them containing a single integer — the number of pairs (i, j) such that item i is at least as valuable as item j after the first k updates.

input
2 3 3 1 2 3 3 2 1 2 1 3 2 1 3 1 1 2 4 2 3 2 4 1 3 1 4 2 1 1 3 2 1 3

output
6 9 9 12 11

In the first test case, after the first update, both permutations are the identity permutations and thus the only valid pairs are $(1, 1)$, $(2, 1)$, $(2, 2)$, $(3, 1)$, $(3, 2)$, and $(3, 3)$. After the second update, it is possible to get any item starting with any item. For instance, you can get item 3 from item 1 using the second trader since $s_1 = 3 > 1 = s_3$. After the third update, it is still possible to get any item starting with any item. For instance, if you start with item 2 and want to get item 1, you can first exchange your item 2 for item 3 using the second trader, and then exchange item 3 for item 1 using the first trader.

G. Modular Tetration

2 seconds, 256 megabytes

For a positive integer a, we define a recurrence $\{b_n\}_{n \ge 0}$ as $b_n = a^{b_{n-1}}$, with $b_0 = 1$.

We say that a positive integer a is *m-tetrative* if the sequence b stabilizes to 1 modulo m, that is, there exists $N \ge 0$ such that $b_n \equiv 1 \pmod m$ for all $n \ge N$.

For a given m, calculate the density of the *m-tetrative* integers. Here, the density of a set S is the limit

$$\lim_{n \to \infty} \frac{|S \cap [1, 2, \ldots, n]|}{n}.$$

Informally, it is the "proportion" of positive integers that are *m-tetrative*.

It can be proven (under the constraints of this problem) that the density is well-defined and is always a rational number, whose denominator is not divisible by $998\,244\,353$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The number m will be given to you as a product of three integers x, y, and z. The first and only line of each test case contains three integers x, y, z ($1 \le x, y, z \le 10^6$, $m = xyz \ge 2$).

Output

For each test case, print the density of *m-tetrative* integers (a such that its corresponding sequence b_n converges to 1 modulo m), modulo $998\,244\,353$.

Formally, let $M = 998\,244\,353$. It can be shown that the exact answer can be expressed as an irreducible fraction $\frac{p}{q}$, where p and q are integers and $q \not\equiv 0 \pmod M$. Output the integer equal to $p \cdot q^{-1} \bmod M$. In other words, output such an integer x that $0 \le x < M$ and $x \cdot q \equiv p \pmod M$.

input
5 5 1 1 5 2 1 23 1 1 10 10 2 9 3 37

output
499122177 299473306 893685162 913393583 705965601

In the first test case, $m = 5$. For example, $a = 1$ is *5-tetrative* since $b_n = 1$ for all n. For $a = 8$, the sequence b is $[1, 8, 8^8, 8^{(8^8)}, \ldots]$, which is $[1, 3, 1, 1, \ldots]$ if we look at it modulo 5. It can be proven that the sequence is always $1 \pmod 5$ for big enough n, so 8 is *5-tetrative*. For $a = 10$, the sequence b is $[1, 10, 10^{10}, 10^{(10^{10})}, \ldots]$, which is $[1, 0, 0, 0, \ldots]$ if we look at it modulo 5. It can be proven that the sequence is always $0 \pmod 5$ for big enough n, so 10 is not *5-tetrative*. The answer for the first test case is $\frac{1}{2}$.

In the second test case, $m = 10$ and the answer is $\frac{1}{10}$.

In the third test case, $m = 23$ and the answer is $\frac{63}{506}$.

In the fourth test case, $m = 200$ and the answer is $\frac{1}{200}$.

In the fifth test case, $m = 999$ and the answer is $\frac{1}{222}$.

H. Maxflow GCD Coloring

2 seconds, 256 megabytes

Consider an undirected graph G with n vertices and positive integer capacities on its edges. We denote by $\text{maxflow}(u, v)$ the value of the maximum flow in the graph from source u to sink v. We say that such a graph G is *good* if there exists a positive integer $d \geq 2$ such that d divides each of the $n \cdot (n-1)$ values of $\text{maxflow}(u, v)$ for all pairs (u, v) of distinct vertices of G. Note that, in particular, any graph without edges is good.

You are given a graph, and you have to color its vertices so that, for each color, the subgraph induced* by the vertices of that color is good. Find such a coloring with the **minimum** possible number of colors.

* Subgraph induced by a set of vertices S is another graph, whose vertices are S and whose edges are **all** of the edges, from the original graph, connecting pairs of vertices in S.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 1000$). The description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \leq n \leq 50$, $0 \leq m \leq \frac{n(n-1)}{2}$), the number of vertices and the number of edges of the graph.

The next m lines each contain three integers u, v, w ($1 \leq u, v \leq n$, $1 \leq w \leq 10^6$), denoting that there is an edge connecting vertices u and v with capacity w. It is guaranteed that there are no self-loops or multiple edges between the same vertices.

It is guaranteed that the sum of n^4 over all test cases does not exceed 50^4.

It can be shown that other limitations imply that the sum of m over all test cases does not exceed $50\,000$, so you don't need to worry about the size of the input.

Output

For each test case, output one integer c, the minimum possible number of colors. Then, output $2c$ lines, describing the coloring. For each $1 \leq i \leq c$, you must output two lines: one line with the number of vertices k_i colored with the i-th color, followed by one line with k_i integers, the vertices colored with the i-th color.

If there are multiple solutions, print any of them.

input

```
2
5 5
1 2 2
2 3 3
3 4 4
4 5 5
5 1 6
6 7
1 2 2
1 3 2
1 4 2
2 5 1
3 5 1
4 5 1
5 6 6
```

output

```
2
2
2 4
3
1 3 5
1
6
1 2 3 4 5 6
```

Visualizer link

In the answer to the first test case, the subgraph induced by the first color has no edges, and the subgraph induced by the second color has only one edge with capacity 6; therefore, both subgraphs are good.

In the second test case, the whole graph is good since the value of maxflow between any pair of vertices is divisible by 3.

I1. Longest Increasing Path (Easy Version)

2 seconds, 256 megabytes

This is the easy version of the problem. The difference between the versions is that in this version the second test is $n = 100\,000$. Hacks are not allowed in this problem.

Let's call an integer sequence $a_1, a_2, \ldots, a_n$ *distance-convex* if $|a_i - a_{i-1}| < |a_{i+1} - a_i|$ for every $1 < i < n$. In other words, if you jump through the points $a_1, a_2, \ldots, a_n$ on the number line in this order, each next jump is strictly longer than the previous one.

You are given two integers n and m. Find a *distance-convex* sequence a of length n that contains **at most m distinct values**. In terms of the jump sequence, that would mean that you need to make $(n-1)$ jumps of increasing lengths while visiting at most m distinct points. $-10^{18} \leq a_i \leq 10^{18}$ should hold for all $1 \leq i \leq n$.

Input

Input consists of two integers n and m on a single line.

There are only 2 tests for this problem.

- $n = 8$, $m = 6$;
- $n = 100\,000$, $m = 15\,000$.

It is guaranteed that an answer exists for both tests.
Hacks are not allowed in this problem.

Output

Print a distance-convex sequence $a_1, a_2, \ldots, a_n$ that contains at most m distinct values. $-10^{18} \leq a_i \leq 10^{18}$ should hold for all $1 \leq i \leq n$.

If there are multiple solutions, print any of them.

input

```
8 6
```

output

```
1 1 3 6 10 3 11 1
```

The sequence $[1, 1, 3, 6, 10, 3, 11, 1]$ has length $n = 8$ and contains 5 different values, which is less than or equal to $m = 6$. The differences between consecutive values are $0, 2, 3, 4, 7, 8, 10$, a strictly increasing sequence.

$[1, 1, 2, 4, 8, 16, 32, 1]$ would be another valid answer.

I2. Longest Increasing Path (Hard Version)

2 seconds, 256 megabytes

This is the hard version of the problem. The difference between the versions is that in this version the second test is $n = 300\,000$. Hacks are not allowed in this problem.

Let's call an integer sequence $a_1, a_2, \ldots, a_n$ *distance-convex* if $|a_i - a_{i-1}| < |a_{i+1} - a_i|$ for every $1 < i < n$. In other words, if you jump through the points $a_1, a_2, \ldots, a_n$ in this order, each next jump is strictly longer than the previous one.

You are given two integers n and m. Find a *distance-convex* sequence a of length n that contains **at most m distinct values**. In terms of the jump sequence, that would mean that you need to make $(n-1)$ jumps of increasing lengths while visiting at most m distinct points.
$-10^{18} \le a_i \le 10^{18}$ should hold for all $1 \le i \le n$.

Input

Input consists of two integers n and m.

There are only 2 tests for this problem.

- $n = 8$, $m = 6$;
- $n = 300\,000$, $m = 15\,000$.

It is guaranteed that an answer exists for both tests.
Hacks are not allowed in this problem.

Output

Print a distance-convex sequence $a_1, a_2, \ldots, a_n$ that contains at most m distinct values. $-10^{18} \le a_i \le 10^{18}$ should hold for all $1 \le i \le n$.

If there are multiple solutions, print any of them.

input
8 6
output
1 1 3 6 10 3 11 1

The sequence $[1, 1, 3, 6, 10, 3, 11, 1]$ has length $n = 8$ and contains 5 different values, which is less than or equal to $m = 6$. The differences between consecutive values are $0, 2, 3, 4, 7, 8, 10$, a strictly increasing sequence.

$[1, 1, 2, 4, 8, 16, 32, 1]$ would be another valid answer.