Codeforces Round #624 (Div. 3)

A. Add Odd or Subtract Even

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

You are given two positive integers a and b.

In one move, you can **change** a in the following way:

- Choose any positive **odd** integer x ($x > 0$) and replace a with $a + x$;
- choose any positive **even** integer y ($y > 0$) and replace a with $a - y$.

You can perform as many such operations as you want. You can choose the same numbers x and y in different moves.

Your task is to find the minimum number of moves required to obtain b from a. It is guaranteed that you can always obtain b from a.

You have to answer t independent test cases.

Input

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

Then t test cases follow. Each test case is given as two space-separated integers a and b ($1 \le a, b \le 10^9$).

Output

For each test case, print the answer — the minimum number of moves required to obtain b from a if you can perform any number of moves described in the problem statement. It is guaranteed that you can always obtain b from a.

Example

input
5 2 3 10 10 2 4 7 4 9 3
output
1 0 2 2 1

Note

In the first test case, you can just add 1.

In the second test case, you don't need to do anything.

In the third test case, you can add 1 two times.

In the fourth test case, you can subtract 4 and add 1.

In the fifth test case, you can just subtract 6.

B. WeirdSort

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

You are given an array a of length n.

You are also given a set of **distinct** positions $p_1, p_2, \ldots, p_m$, where $1 \le p_i < n$. The position p_i means that you can swap elements $a[p_i]$ and $a[p_i + 1]$. You can apply this operation any number of times for each of the given **positions**.

Your task is to determine if it is possible to sort the initial array in non-decreasing order ($a_1 \le a_2 \le \cdots \le a_n$) using only allowed swaps.

For example, if $a = [3, 2, 1]$ and $p = [1, 2]$, then we can first swap elements $a[2]$ and $a[3]$ (because position 2 is contained in the given set p). We get the array $a = [3, 1, 2]$. Then we swap $a[1]$ and $a[2]$ (1 is also contained in p). We get the array $a = [1, 3, 2]$. Finally, we swap $a[2]$ and $a[3]$ again and get the array $a = [1, 2, 3]$, sorted in non-decreasing order.

You can see that if $a = [4, 1, 2, 3]$ and $p = [3, 2]$ then you cannot sort the array.

You have to answer t independent test cases.

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

Then t test cases follow. The first line of each test case contains two integers n and m ($1 \le m < n \le 100$) — the number of elements in a and the number of elements in p. The second line of the test case contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 100$). The third line of the test case contains m integers $p_1, p_2, \dots, p_m$ ($1 \le p_i < n$, all p_i are distinct) — the set of positions described in the problem statement.

Output
For each test case, print the answer — "YES" (without quotes) if you can sort the initial array in non-decreasing order ($a_1 \le a_2 \le \dots \le a_n$) using only allowed swaps. Otherwise, print "NO".

Example

input
6 3 2 3 2 1 1 2 4 2 4 1 2 3 3 2 5 1 1 2 3 4 5 1 4 2 2 1 4 3 1 3 4 2 4 3 2 1 1 3 5 2 2 1 2 3 3 1 4
output
YES NO YES YES NO YES

C. Perform the Combo

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

You want to perform the combo on your opponent in one popular fighting game. The combo is the string s consisting of n lowercase Latin letters. To perform the combo, you have to press all buttons in the order they appear in s. I.e. if s ="abca" then you have to press 'a', then 'b', 'c' and 'a' again.

You know that you will spend m wrong tries to perform the combo and during the i-th try you will make a mistake right after p_i-th button ($1 \le p_i < n$) (i.e. you will press first p_i buttons right and start performing the combo from the beginning). It is guaranteed that during the $m + 1$-th try you press all buttons right and finally perform the combo.

I.e. if s ="abca", $m = 2$ and $p = [1, 3]$ then the sequence of pressed buttons will be 'a' (**here** you're making a mistake and start performing the combo from the beginning), 'a', 'b', 'c', (**here** you're making a mistake and start performing the combo from the beginning), 'a' (**note that at this point you will not perform the combo because of the mistake**), 'b', 'c', 'a'.

Your task is to calculate for each button (letter) the number of times you'll press it.

You have to answer t independent test cases.

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

Then t test cases follow.

The first line of each test case contains two integers n and m ($2 \le n \le 2 \cdot 10^5$, $1 \le m \le 2 \cdot 10^5$) — the length of s and the number of tries correspondingly.

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

The third line of each test case contains m integers $p_1, p_2, \ldots, p_m$ ($1 \le p_i < n$) — the number of characters pressed right during the i-th try.

It is guaranteed that the sum of n and the sum of m both does not exceed $2 \cdot 10^5$ ($\sum n \le 2 \cdot 10^5$, $\sum m \le 2 \cdot 10^5$).

It is guaranteed that the answer for each letter does not exceed $2 \cdot 10^9$.

Output
For each test case, print the answer — 26 integers: the number of times you press the button 'a', the number of times you press the button 'b', ..., the number of times you press the button 'z'.

Example

input
3 4 2 abca 1 3 10 5 codeforces 2 8 3 2 9 26 10 qwertyuioplkjhgfdsazxcvbnm 20 10 1 2 3 5 10 5 9 4
output
4 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 4 5 3 0 0 0 0 0 0 0 0 0 9 0 0 3 1 0 0 0 0 0 0 0 0 2 1 1 2 9 2 2 2 5 2 2 2 1 1 5 4 11 8 2 7 5 1 10 1 5 2

Note
The first test case is described in the problem statement. Wrong tries are "a", "abc" and the final try is "abca". The number of times you press 'a' is 4, 'b' is 2 and 'c' is 2.

In the second test case, there are five wrong tries: "co", "codeforc", "cod", "co", "codeforce" and the final try is "codeforces". The number of times you press 'c' is 9, 'd' is 4, 'e' is 5, 'f' is 3, 'o' is 9, 'r' is 3 and 's' is 1.

D. Three Integers

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

You are given three integers $a \le b \le c$.

In one move, you can add $+1$ or -1 to **any** of these integers (i.e. increase or decrease any number by one). You can perform such operation any (possibly, zero) number of times, you can even perform this operation several times with one number. **Note that you cannot make non-positive numbers using such operations**.

You have to perform the minimum number of such operations in order to obtain three integers $A \le B \le C$ such that B is divisible by A and C is divisible by B.

You have to answer t independent test cases.

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

The next t lines describe test cases. Each test case is given on a separate line as three space-separated integers a, b and c ($1 \le a \le b \le c \le 10^4$).

Output
For each test case, print the answer. In the first line print res — the minimum number of operations you have to perform to obtain three integers $A \le B \le C$ such that B is divisible by A and C is divisible by B. On the second line print **any** suitable triple A, B and C.

Example

input
8 1 2 3 123 321 456 5 10 15 15 18 21 100 100 101 1 22 29 3 19 38 6 30 46
output

```
1
1 1 3
102
114 228 456
4
4 8 16
6
18 18 18
1
100 100 100
7
1 22 22
2
1 19 38
8
6 24 48
```

E. Construct the Binary Tree

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

You are given two integers n and d. You need to construct a rooted binary tree consisting of n vertices with a root at the vertex 1 and the sum of depths of all vertices equals to d.

A tree is a connected graph without cycles. A rooted tree has a special vertex called the root. A parent of a vertex v is the last different from v vertex on the path from the root to the vertex v. The depth of the vertex v is the length of the path from the root to the vertex v. Children of vertex v are all vertices for which v is the parent. The binary tree is such a tree that no vertex has more than 2 children.

You have to answer t independent test cases.

Input

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

The only line of each test case contains two integers n and d ($2 \le n, d \le 5000$) — the number of vertices in the tree and the required sum of depths of all vertices.

It is guaranteed that the sum of n and the sum of d both does not exceed 5000 ($\sum n \le 5000, \sum d \le 5000$).

Output

For each test case, print the answer.

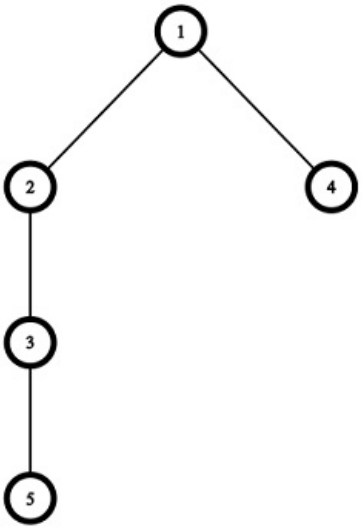
If it is impossible to construct such a tree, print "NO" (without quotes) in the first line. Otherwise, print "{YES}" in the first line. Then print $n - 1$ integers $p_2, p_3, \ldots, p_n$ in the second line, where p_i is the parent of the vertex i. Note that the sequence of parents you print should describe some binary tree.

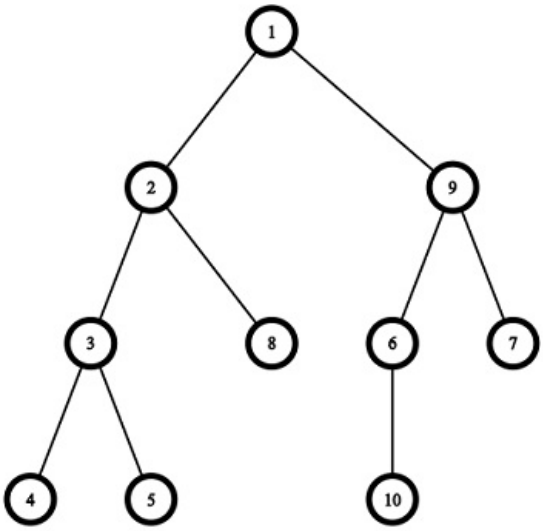
Example

input
3 5 7 10 19 10 18
output
YES 1 2 1 3 YES 1 2 3 3 9 9 2 1 6 NO

Note

Pictures corresponding to the first and the second test cases of the example:





F. Moving Points

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

There are n points on a coordinate axis OX. The i-th point is located at the integer point x_i and has a speed v_i. It is guaranteed that no two points occupy the same coordinate. All n points move with the constant speed, the coordinate of the i-th point at the moment t (t **can be non-integer**) is calculated as $x_i + t \cdot v_i$.

Consider two points i and j. Let $d(i, j)$ be the minimum possible distance between these two points over any possible moments of time (even **non-integer**). It means that if two points i and j coincide at some moment, the value $d(i, j)$ will be 0.

Your task is to calculate the value $\sum\limits_{1 \le i < j \le n} d(i, j)$ (the sum of minimum distances over all pairs of points).

Input

The first line of the input contains one integer n ($2 \le n \le 2 \cdot 10^5$) — the number of points.

The second line of the input contains n integers $x_1, x_2, \ldots, x_n$ ($1 \le x_i \le 10^8$), where x_i is the initial coordinate of the i-th point. It is guaranteed that all x_i are distinct.

The third line of the input contains n integers $v_1, v_2, \ldots, v_n$ ($-10^8 \le v_i \le 10^8$), where v_i is the speed of the i-th point.

Output

Print one integer — the value $\sum\limits_{1 \le i < j \le n} d(i, j)$ (the sum of minimum distances over all pairs of points).

Examples

input
3 1 3 2 -100 2 3
output

3

input
5 2 1 4 3 5 2 2 2 3 4
output
19

input
2 2 1 -3 0
output
0