Codeforces Round #667 (Div. 3)

A. Yet Another Two Integers Problem

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

You are given two integers a and b.

In one move, you can choose some **integer** k from 1 to 10 and add it to a or subtract it from a. In other words, you choose an integer $k \in [1; 10]$ and perform $a := a + k$ or $a := a - k$. You may use **different** values of k in different moves.

Your task is to find the **minimum** number of moves required to obtain b from a.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 2 \cdot 10^4$) — the number of test cases. Then t test cases follow.

The only line of the test case contains two 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.

Example

input
6 5 5 13 42 18 4 1337 420 123456789 1000000000 100500 9000
output
0 3 2 92 87654322 9150

Note

In the first test case of the example, you don't need to do anything.

In the second test case of the example, the following sequence of moves can be applied: $13 \to 23 \to 32 \to 42$ (add 10, add 9, add 10).

In the third test case of the example, the following sequence of moves can be applied: $18 \to 10 \to 4$ (subtract 8, subtract 6).

B. Minimum Product

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

You are given four integers a, b, x and y. Initially, $a \ge x$ and $b \ge y$. You can do the following operation **no more than** n times:

- Choose either a or b and decrease it by one. However, as a result of this operation, value of a cannot become less than x, and value of b cannot become less than y.

Your task is to find the **minimum** possible product of a and b ($a \cdot b$) you can achieve by applying the given operation no more than n times.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 2 \cdot 10^4$) — the number of test cases. Then t test cases follow.

The only line of the test case contains five integers a, b, x, y and n ($1 \le a, b, x, y, n \le 10^9$). Additional constraint on the input:

$a \geq x$ and $b \geq y$ always holds.

Output
For each test case, print one integer: the **minimum** possible product of a and b ($a \cdot b$) you can achieve by applying the given operation no more than n times.

Example

input
7 10 10 8 5 3 12 8 8 7 2 12343 43 4543 39 123212 1000000000 1000000000 1 1 1 1000000000 1000000000 1 1 1000000000 10 11 2 1 5 10 11 9 1 10
output
70 77 177177 999999999000000000 999999999 55 10

Note
In the first test case of the example, you need to decrease b three times and obtain $10 \cdot 7 = 70$.

In the second test case of the example, you need to decrease a one time, b one time and obtain $11 \cdot 7 = 77$.

In the sixth test case of the example, you need to decrease a five times and obtain $5 \cdot 11 = 55$.

In the seventh test case of the example, you need to decrease b ten times and obtain $10 \cdot 1 = 10$.

C. Yet Another Array Restoration

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

We have a secret array. You don't know this array and you have to restore it. However, you know some facts about this array:

- The array consists of n **distinct positive** (greater than 0) integers.
- The array contains two elements x and y (these elements are **known** for you) such that $x < y$.
- If you sort the array in increasing order (such that $a_1 < a_2 < \ldots < a_n$), differences between all adjacent (consecutive) elements are equal (i.e. $a_2 - a_1 = a_3 - a_2 = \ldots = a_n - a_{n-1}$).

It can be proven that such an array always exists under the constraints given below.

Among all possible arrays that satisfy the given conditions, we ask you to restore one which has the **minimum possible** maximum element. In other words, you have to minimize $\max(a_1, a_2, \ldots, a_n)$.

You have to answer t independent test cases.

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

The only line of the test case contains three integers n, x and y ($2 \leq n \leq 50$; $1 \leq x < y \leq 50$) — the length of the array and two elements that are present in the array, respectively.

Output
For each test case, print the answer: n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$), where a_i is the i-th element of the required array. If there are several answers, you can print any (it also means that the order of elements doesn't matter).

It can be proven that such an array always exists under the given constraints.

Example

input
5 2 1 49 5 20 50 6 20 50 5 3 8 9 13 22
output
1 49 20 40 30 50 10

```
26 32 20 38 44 50
8 23 18 13 3
1 10 13 4 19 22 25 16 7
```

D. Decrease the Sum of Digits

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

You are given a positive integer n. In one move, you can increase n by one (i.e. make $n := n + 1$). Your task is to find the minimum number of moves you need to perform in order to make the sum of digits of n be less than or equal to s.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 2 \cdot 10^4$) — the number of test cases. Then t test cases follow.

The only line of the test case contains two integers n and s ($1 \le n \le 10^{18}$; $1 \le s \le 162$).

Output

For each test case, print the answer: the minimum number of moves you need to perform in order to make the sum of digits of n be less than or equal to s.

Example

input
5 2 1 1 1 500 4 217871987498122 10 100000000000000001 1
output
8 0 500 2128012501878 899999999999999999

E. Two Platforms

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

There are n points on a plane. The i-th point has coordinates (x_i, y_i). You have two horizontal platforms, both of length k. Each platform can be placed anywhere on a plane but it should be placed **horizontally** (on the same y-coordinate) and have **integer borders**. If the left border of the platform is (x, y) then the right border is $(x + k, y)$ and all points between borders (including borders) belong to the platform.

Note that platforms can share common points (overlap) and it is not necessary to place both platforms on the same y-coordinate.

When you place both platforms on a plane, all points start falling down decreasing their y-coordinate. If a point collides with some platform at some moment, the point stops and is **saved**. Points which never collide with any platform are lost.

Your task is to find the maximum number of points you can **save** if you place both platforms optimally.

You have to answer t independent test cases.

For better understanding, please read the **Note** section below to see a picture for the first test case.

Input

The first line of the input contains one integer t ($1 \le t \le 2 \cdot 10^4$) — the number of test cases. Then t test cases follow.

The first line of the test case contains two integers n and k ($1 \le n \le 2 \cdot 10^5$; $1 \le k \le 10^9$) — the number of points and the length of each platform, respectively. The second line of the test case contains n integers $x_1, x_2, \ldots, x_n$ ($1 \le x_i \le 10^9$), where x_i is x-coordinate of the i-th point. The third line of the input contains n integers $y_1, y_2, \ldots, y_n$ ($1 \le y_i \le 10^9$), where y_i is y-coordinate of the i-th point. All points are distinct (there is no pair $1 \le i < j \le n$ such that $x_i = x_j$ and $y_i = y_j$).

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

Output

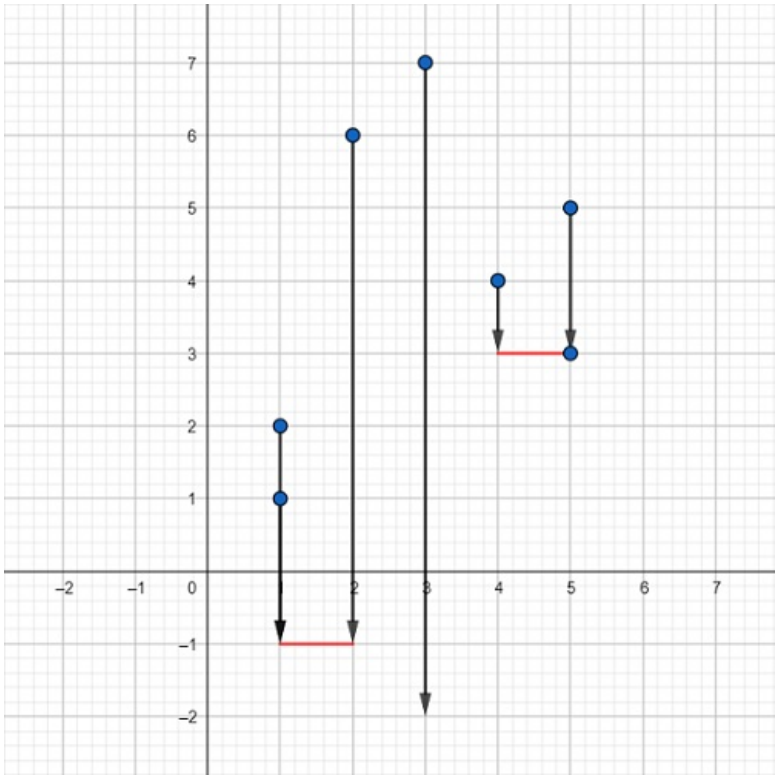
For each test case, print the answer: the maximum number of points you can save if you place both platforms optimally.

Example

input
4 7 1 1 5 2 3 1 5 4 1 3 6 7 2 5 4 1 1 1000000000 1000000000 5 10 10 7 5 15 8 20 199 192 219 1904 10 10 15 19 8 17 20 10 9 2 10 19 12 13 6 17 1 14 7 9 19 3
output
6 1 5 10

Note

The picture corresponding to the first test case of the example:



Blue dots represent the points, red segments represent the platforms. One of the possible ways is to place the first platform between points $(1, -1)$ and $(2, -1)$ and the second one between points $(4, 3)$ and $(5, 3)$. Vectors represent how the points will fall down. As you can see, the only point we can't save is the point $(3, 7)$ so it falls down infinitely and will be lost. It can be proven that we can't achieve better answer here. Also note that the point $(5, 3)$ doesn't fall at all because it is already on the platform.

F. Subsequences of Length Two

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

You are given two strings s and t consisting of lowercase Latin letters. The length of t is 2 (i.e. this string consists only of two characters).

In one move, you can choose **any** character of s and replace it with **any** lowercase Latin letter. More formally, you choose some i and replace s_i (the character at the position i) with some character from 'a' to 'z'.

You want to do **no more than** k replacements in such a way that **maximizes** the number of occurrences of t in s as a **subsequence**.

Recall that a subsequence is a sequence that can be derived from the given sequence by deleting zero or more elements without changing the order of the remaining elements.

Input

The first line of the input contains two integers n and k ($2 \le n \le 200$; $0 \le k \le n$) — the length of s and the maximum number of

moves you can make. The second line of the input contains the string s consisting of n lowercase Latin letters. The third line of the input contains the string t consisting of two lowercase Latin letters.

Output

Print one integer — the maximum possible number of occurrences of t in s as a **subsequence** if you replace no more than k characters in s optimally.

Examples

input
4 2 bbaa ab
output
3

input
7 3 asddsaf sd
output
10

input
15 6 qwertyhgfdsazxc qa
output
16

input
7 2 abacaba aa
output
15

Note

In the first example, you can obtain the string "abab" replacing s_1 with 'a' and s_4 with 'b'. Then the answer is 3.

In the second example, you can obtain the string "ssddsdd" and get the answer 10.

In the fourth example, you can obtain the string "aaacaaa" and get the answer 15.