Educational Codeforces Round 126 (Rated for Div. 2)

A. Array Balancing

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

You are given two arrays of length n: $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$.

You can perform the following operation any number of times:

1. Choose integer index i ($1 \le i \le n$);
2. Swap a_i and b_i.

What is the minimum possible sum $|a_1 - a_2| + |a_2 - a_3| + \cdots + |a_{n-1} - a_n| + |b_1 - b_2| + |b_2 - b_3| + \cdots + |b_{n-1} - b_n|$ (in other words, $\sum\limits_{i=1}^{n-1} (|a_i - a_{i+1}| + |b_i - b_{i+1}|)$) you can achieve after performing several (possibly, zero) operations?

Input

The first line contains a single integer t ($1 \le t \le 4000$) — the number of test cases. Then, t test cases follow.

The first line of each test case contains the single integer n ($2 \le n \le 25$) — the length of arrays a and b.

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

The third line of each test case contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10^9$) — the array b.

Output

For each test case, print one integer — the minimum possible sum $\sum\limits_{i=1}^{n-1} (|a_i - a_{i+1}| + |b_i - b_{i+1}|)$.

Example

input
3 4 3 3 10 10 10 10 3 3 5 1 2 3 4 5 6 7 8 9 10 6 72 101 108 108 111 44 10 87 111 114 108 100
output
0 8 218

Note

In the first test case, we can, for example, swap a_3 with b_3 and a_4 with b_4. We'll get arrays $a = [3, 3, 3, 3]$ and $b = [10, 10, 10, 10]$ with sum $3 \cdot |3 - 3| + 3 \cdot |10 - 10| = 0$.

In the second test case, arrays already have minimum sum (described above) equal to $|1 - 2| + \cdots + |4 - 5| + |6 - 7| + \cdots + |9 - 10| = 4 + 4 = 8$.

In the third test case, we can, for example, swap a_5 and b_5.

B. Getting Zero

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

Suppose you have an integer v. In one operation, you can:

- either set $v = (v + 1) \bmod 32768$
- or set $v = (2 \cdot v) \bmod 32768$.

You are given n integers $a_1, a_2, \ldots, a_n$. What is the minimum number of operations you need to make each a_i equal to 0?

Input
The first line contains the single integer n ($1 \le n \le 32768$) — the number of integers.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < 32768$).

Output
Print n integers. The i-th integer should be equal to the minimum number of operations required to make a_i equal to 0.

Example

input
4 19 32764 10240 49
output
14 4 4 15

Note
Let's consider each a_i:

- $a_1 = 19$. You can, firstly, increase it by one to get 20 and then multiply it by two 13 times. You'll get 0 in $1 + 13 = 14$ steps.
- $a_2 = 32764$. You can increase it by one 4 times: $32764 \to 32765 \to 32766 \to 32767 \to 0$.
- $a_3 = 10240$. You can multiply it by two 4 times: $10240 \to 20480 \to 8192 \to 16384 \to 0$.
- $a_4 = 49$. You can multiply it by two 15 times.

C. Water the Trees

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n trees in a park, numbered from 1 to n. The initial height of the i-th tree is h_i.

You want to water these trees, so they all grow to the **same** height.

The watering process goes as follows. You start watering trees at day 1. During the j-th day you can:

- Choose a tree and water it. If the day is odd (e.g. $1, 3, 5, 7, \ldots$), then the height of the tree increases by 1. If the day is even (e.g. $2, 4, 6, 8, \ldots$), then the height of the tree increases by 2.
- Or skip a day without watering any tree.

Note that you can't water more than one tree in a day.

Your task is to determine the **minimum** number of days required to water the trees so they grow to the same height.

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.

The first line of the test case contains one integer n ($1 \le n \le 3 \cdot 10^5$) — the number of trees.

The second line of the test case contains n integers $h_1, h_2, \ldots, h_n$ ($1 \le h_i \le 10^9$), where h_i is the height of the i-th tree.

It is guaranteed that the sum of n over all test cases does not exceed $3 \cdot 10^5$ ($\sum n \le 3 \cdot 10^5$).

Output
For each test case, print one integer — the **minimum** number of days required to water the trees, so they grow to the same height.

Example

input
3 3 1 2 4 5 4 4 3 5 5 7 2 5 4 8 3 7 4
output
4 3 16

Note
Consider the first test case of the example. The initial state of the trees is $[1, 2, 4]$.

1. During the first day, let's water the first tree, so the sequence of heights becomes $[2, 2, 4]$;
2. during the second day, let's water the second tree, so the sequence of heights becomes $[2, 4, 4]$;
3. let's skip the third day;
4. during the fourth day, let's water the first tree, so the sequence of heights becomes $[4, 4, 4]$.

Thus, the answer is 4.

D. Progressions Covering

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

You are given two arrays: an array a consisting of n zeros and an array b consisting of n integers.

You can apply the following operation to the array a an arbitrary number of times: choose some subsegment of a of length k and add the arithmetic progression $1, 2, \ldots, k$ to this subsegment — i. e. add 1 to the first element of the subsegment, 2 to the second element, and so on. The chosen subsegment should be inside the borders of the array a (i.e., if the left border of the chosen subsegment is l, then the condition $1 \le l \le l + k - 1 \le n$ should be satisfied). Note that the progression added is always $1, 2, \ldots, k$ but not the $k, k - 1, \ldots, 1$ or anything else (i.e., the leftmost element of the subsegment always increases by 1, the second element always increases by 2 and so on).

Your task is to find the **minimum** possible number of operations required to satisfy the condition $a_i \ge b_i$ for each i from 1 to n. Note that the condition $a_i \ge b_i$ should be satisfied for all elements at once.

Input

The first line of the input contains two integers n and k ($1 \le k \le n \le 3 \cdot 10^5$) — the number of elements in both arrays and the length of the subsegment, respectively.

The second line of the input contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10^{12}$), where b_i is the i-th element of the array b.

Output

Print one integer — the **minimum** possible number of operations required to satisfy the condition $a_i \ge b_i$ for each i from 1 to n.

Examples

input
3 3 5 4 6
output
5

input
6 3 1 2 3 2 2 3
output
3

input
6 3 1 2 4 1 2 3
output
3

input
7 3 50 17 81 25 42 39 96
output
92

Note

Consider the first example. In this test, we don't really have any choice, so we need to add at least five progressions to make the first element equals 5. The array a becomes $[5, 10, 15]$.

Consider the second example. In this test, let's add one progression on the segment $[1; 3]$ and two progressions on the segment $[4; 6]$. Then, the array a becomes $[1, 2, 3, 2, 4, 6]$.

E. Narrow Components

time limit per test: 2 seconds

memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a matrix a, consisting of 3 rows and n columns. Each cell of the matrix is either free or taken.

A free cell y is reachable from a free cell x if at least one of these conditions hold:

- x and y share a side;
- there exists a free cell z such that z is reachable from x and y is reachable from z.

A connected component is a set of free cells of the matrix such that all cells in it are reachable from one another, but adding any other free cell to the set violates this rule.

You are asked q queries about the matrix. Each query is the following:

- l r — count the number of connected components of the matrix, consisting of columns from l to r of the matrix a, inclusive.

Print the answers to all queries.

Input
The first line contains an integer n ($1 \le n \le 5 \cdot 10^5$) — the number of columns of matrix a.

The i-th of the next three lines contains a description of the i-th row of the matrix a — a string, consisting of n characters. Each character is either 1 (denoting a free cell) or 0 (denoting a taken cell).

The next line contains an integer q ($1 \le q \le 3 \cdot 10^5$) — the number of queries.

The j-th of the next q lines contains two integers l_j and r_j ($1 \le l_j \le r_j \le n$) — the description of the j-th query.

Output
Print q integers — the j-th value should be equal to the number of the connected components of the matrix, consisting of columns from l_j to r_j of the matrix a, inclusive.

Example

input
12 100101011101 110110010110 010001011101 8 1 12 1 1 1 2 9 9 8 11 9 12 11 12 4 6
output
7 1 1 2 1 3 3 3

F. Teleporters

time limit per test: 7 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

There are $n + 1$ teleporters on a straight line, located in points 0, a_1, a_2, a_3, ..., a_n. It's possible to teleport from point x to point y if there are teleporters in **both** of those points, and it costs $(x - y)^2$ energy.

You want to install some additional teleporters so that it is possible to get from the point 0 to the point a_n (possibly through some other teleporters) spending **no more** than m energy in total. Each teleporter you install must be located in an **integer point**.

What is the minimum number of teleporters you have to install?

Input
The first line contains one integer n ($1 \le n \le 2 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_1 < a_2 < a_3 < \cdots < a_n \le 10^9$).

The third line contains one integer m ($a_n \le m \le 10^{18}$).

Output

Print one integer — the minimum number of teleporters you have to install so that it is possible to get from 0 to a_n spending at most m energy. It can be shown that it's always possible under the constraints from the input format.

Examples

input
2 1 5 7
output
2

input
2 1 5 6
output
3

input
1 5 5
output
4

input
1 1000000000 1000000043
output
999999978