Educational Codeforces Round 92 (Rated for Div. 2)

A. LCM Problem

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

Let $LCM(x, y)$ be the minimum positive integer that is divisible by both x and y. For example, $LCM(13, 37) = 481$, $LCM(9, 6) = 18$.

You are given two integers l and r. Find two integers x and y such that $l \le x < y \le r$ and $l \le LCM(x, y) \le r$.

Input
The first line contains one integer t ($1 \le t \le 10000$) — the number of test cases.

Each test case is represented by one line containing two integers l and r ($1 \le l < r \le 10^9$).

Output
For each test case, print two integers:

- if it is impossible to find integers x and y meeting the constraints in the statement, print two integers equal to -1;
- otherwise, print the values of x and y (if there are multiple valid answers, you may print any of them).

Example

input
4 1 1337 13 69 2 4 88 89
output
6 7 14 21 2 4 -1 -1

B. Array Walk

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

You are given an array $a_1, a_2, \ldots, a_n$, consisting of n **positive** integers.

Initially you are standing at index 1 and have a score equal to a_1. You can perform two kinds of moves:

1. move right — go from your current index x to $x + 1$ and add a_{x+1} to your score. This move can only be performed if $x < n$.
2. move left — go from your current index x to $x - 1$ and add a_{x-1} to your score. This move can only be performed if $x > 1$. **Also, you can't perform two or more moves to the left in a row.**

You want to perform **exactly** k moves. Also, there should be no more than z moves to the left among them.

What is the maximum score you can achieve?

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 three integers n, k and z ($2 \le n \le 10^5$, $1 \le k \le n - 1$, $0 \le z \le min(5, k)$) — the number of elements in the array, the total number of moves you should perform and the maximum number of moves to the left you can perform.

The second line of each testcase contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^4$) — the given array.

The sum of n over all testcases does not exceed $3 \cdot 10^5$.

Output
Print t integers — for each testcase output the maximum score you can achieve if you make exactly k moves in total, no more than z of them are to the left and there are no two or more moves to the left in a row.

Example

input
4 5 4 0 1 5 4 3 2 5 4 1 1 5 4 3 2 5 4 4 10 20 30 40 50 10 7 3 4 6 8 2 9 9 7 4 10 9
output
15 19 150 56

Note
In the first testcase you are not allowed to move left at all. So you make four moves to the right and obtain the score $a_1 + a_2 + a_3 + a_4 + a_5$.

In the second example you can move one time to the left. So we can follow these moves: right, right, left, right. The score will be $a_1 + a_2 + a_3 + a_2 + a_3$.

In the third example you can move four times to the left but it's not optimal anyway, you can just move four times to the right and obtain the score $a_1 + a_2 + a_3 + a_4 + a_5$.

C. Good String

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

Let's call *left cyclic shift* of some string $t_1 t_2 t_3 \dots t_{n-1} t_n$ as string $t_2 t_3 \dots t_{n-1} t_n t_1$.

Analogically, let's call *right cyclic shift* of string t as string $t_n t_1 t_2 t_3 \dots t_{n-1}$.

Let's say string t is **good** if its left cyclic shift is equal to its right cyclic shift.

You are given string s which consists of digits 0–9.

What is the minimum number of characters you need to erase from s to make it good?

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

Next t lines contains test cases — one per line. The first and only line of each test case contains string s ($2 \le |s| \le 2 \cdot 10^5$). Each character s_i is digit 0–9.

It's guaranteed that the total length of strings doesn't exceed $2 \cdot 10^5$.

Output
For each test case, print the minimum number of characters you need to erase from s to make it good.

Example

input
3 95831 100120013 252525252525
output
3 5 0

Note
In the first test case, you can erase any 3 characters, for example, the 1-st, the 3-rd, and the 4-th. You'll get string 51 and it is good.

In the second test case, we can erase all characters except 0: the remaining string is 0000 and it's good.

In the third test case, the given string s is already good.

D. Segment Intersections

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

You are given two lists of segments $[al_1, ar_1], [al_2, ar_2], \dots, [al_n, ar_n]$ and $[bl_1, br_1], [bl_2, br_2], \dots, [bl_n, br_n]$.

Initially, all segments $[al_i, ar_i]$ are equal to $[l_1, r_1]$ and all segments $[bl_i, br_i]$ are equal to $[l_2, r_2]$.

In one step, you can choose one segment (either from the first or from the second list) and extend it by 1. In other words, suppose you've chosen segment $[x, y]$ then you can transform it either into $[x - 1, y]$ or into $[x, y + 1]$.

Let's define a total intersection I as the sum of lengths of intersections of the corresponding pairs of segments, i.e. $\sum\limits_{i=1}^{n} \text{intersection\_length}([al_i, ar_i], [bl_i, br_i])$. Empty intersection has length 0 and length of a segment $[x, y]$ is equal to $y - x$.

What is the minimum number of steps you need to make I greater or equal to k?

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

The first line of each test case contains two integers n and k ($1 \le n \le 2 \cdot 10^5$; $1 \le k \le 10^9$) — the length of lists and the minimum required total intersection.

The second line of each test case contains two integers l_1 and r_1 ($1 \le l_1 \le r_1 \le 10^9$) — the segment all $[al_i, ar_i]$ are equal to initially.

The third line of each test case contains two integers l_2 and r_2 ($1 \le l_2 \le r_2 \le 10^9$) — the segment all $[bl_i, br_i]$ are equal to initially.

It's guaranteed that the sum of n doesn't exceed $2 \cdot 10^5$.

Output
Print t integers — one per test case. For each test case, print the minimum number of step you need to make I greater or equal to k.

Example

input
3 3 5 1 2 3 4 2 1000000000 1 1 999999999 999999999 10 3 5 10 7 8
output
7 2000000000 0

Note
In the first test case, we can achieve total intersection 5, for example, using next strategy:

- make $[al_1, ar_1]$ from $[1, 2]$ to $[1, 4]$ in 2 steps;
- make $[al_2, ar_2]$ from $[1, 2]$ to $[1, 3]$ in 1 step;
- make $[bl_1, br_1]$ from $[3, 4]$ to $[1, 4]$ in 2 steps;
- make $[bl_2, br_2]$ from $[3, 4]$ to $[1, 4]$ in 2 steps.

In result, $I = \text{intersection\_length}([al_1, ar_1], [bl_1, br_1]) + \text{intersection\_length}([al_2, ar_2], [bl_2, br_2]) + {} + \text{intersection\_length}([al_3, ar_3], [bl_3, br_3]) = 3 + 2 + 0 = 5$
In the second test case, we can make $[al_1, ar_1] = [0, 1000000000]$ in 1000000000 steps and $[bl_1, br_1] = [0, 1000000000]$ in 1000000000 steps.

In the third test case, the total intersection I is already equal to $10 > 3$, so we don't need to do any steps.

E. Calendar Ambiguity

time limit per test: 2 seconds

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

Berland year consists of m months with d days each. Months are numbered from 1 to m. Berland week consists of w days. The first day of the year is also the first day of the week. Note that the last week of the year might be shorter than w days.

A pair (x, y) such that $x < y$ is ambiguous if day x of month y is the same day of the week as day y of month x.

Count the number of ambiguous pairs.

Input
The first line contains a single integer t ($1 \le t \le 1000$) — the number of testcases.

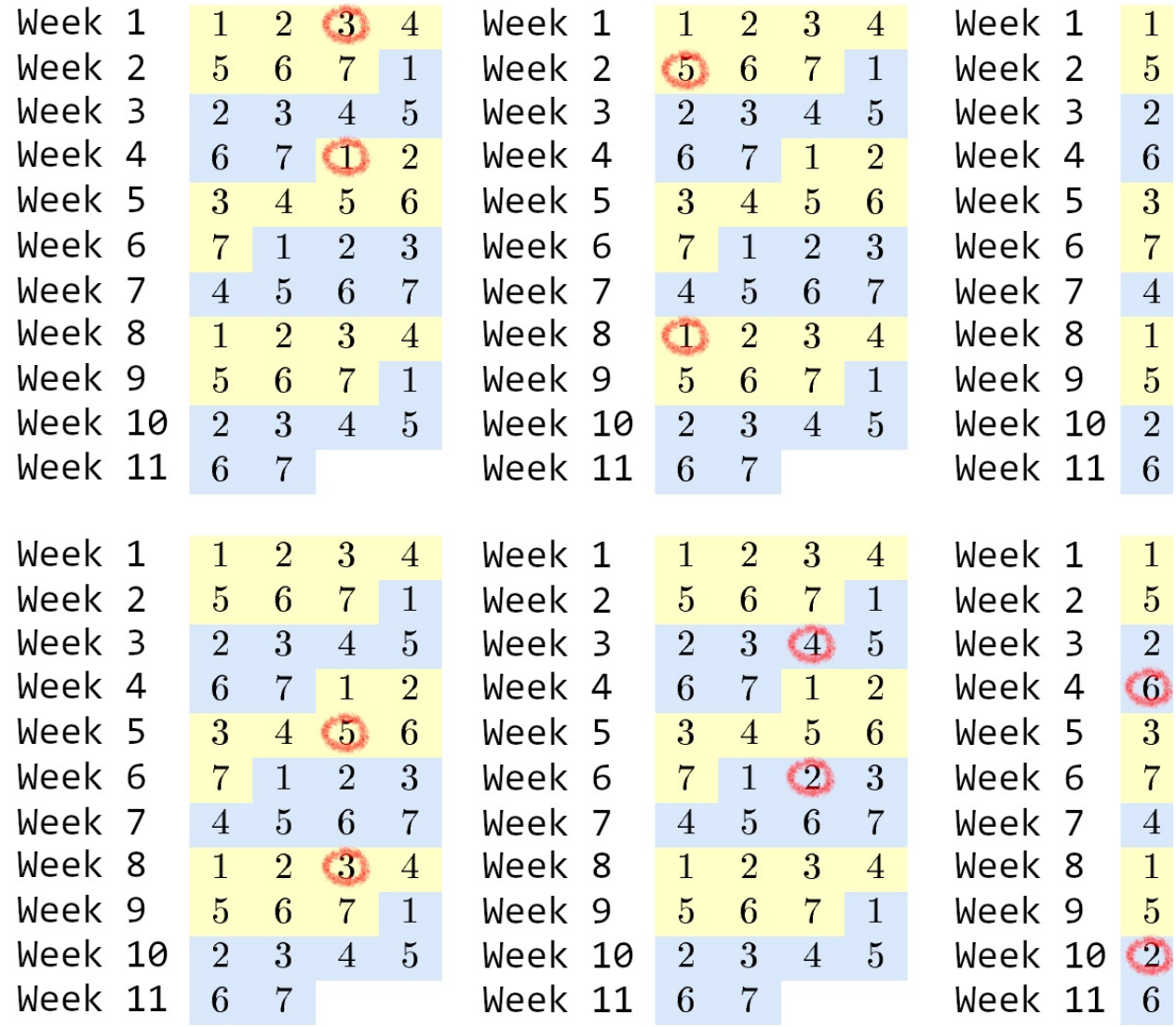
Each of the next t lines contains three integers m, d and w ($1 \le m, d, w \le 10^9$) — the number of months in a year, the number of days in a month and the number of days in a week.

Output
Print t integers — for each testcase output the number of pairs (x, y) such that $x < y$ and day x of month y is the same day of the week as day y of month x.

Example

input
5 6 7 4 10 7 12 12 30 7 1 1 1 3247834 10298779 625324
output
6 9 5 0 116461800

Note
Here are the pairs for the first test case:



F. Bicolored Segments

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

You are given n segments $[l_1, r_1], [l_2, r_2], \ldots, [l_n, r_n]$. Each segment has one of two colors: the i-th segment's color is t_i.

Let's call a pair of segments i and j *bad* if the following two conditions are met:

- $t_i \neq t_j$;
- the segments $[l_i, r_i]$ and $[l_j, r_j]$ intersect, embed or touch, i. e. there exists an integer x such that $x \in [l_i, r_i]$ and $x \in [l_j, r_j]$.

Calculate the maximum number of segments that can be selected from the given ones, so that there is no *bad* pair among the selected ones.

Input
The first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — number of segments.

The next n lines contains three integers l_i, r_i, t_i ($1 \le l_i \le r_i \le 10^9; t_i \in \{1, 2\}$) — description of the i-th segment.

Output
Print the maximum number of segments that can be selected, so that there is no *bad* pair among the selected segments.

Examples

input
3 1 3 1 4 6 2 2 5 1
output
2

input
5 5 8 1 1 3 2 3 4 2 6 6 1 2 10 2
output
4

input
7 19 20 1 13 15 2 6 11 2 4 10 1 14 17 1 13 13 2 5 9 1
output
5

G. Directing Edges

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given an undirected connected graph consisting of n vertices and m edges. k vertices of this graph are special.

You have to direct each edge of this graph or leave it undirected. If you leave the i-th edge undirected, you pay w_i coins, and if you direct it, you don't have to pay for it.

Let's call a vertex *saturated* if it is reachable from each special vertex along the edges of the graph (if an edge is undirected, it can be traversed in both directions). After you direct the edges of the graph (possibly leaving some of them undirected), you receive c_i coins for each saturated vertex i. Thus, your total profit can be calculated as $\sum\limits_{i \in S} c_i - \sum\limits_{j \in U} w_j$, where S is the set of saturated vertices, and U is the set of edges you leave undirected.

For each vertex i, calculate the maximum possible profit you can get if you have to make the vertex i saturated.

Input
The first line contains three integers n, m and k ($2 \le n \le 3 \cdot 10^5$, $n - 1 \le m \le \min(3 \cdot 10^5, \frac{n(n-1)}{2})$, $1 \le k \le n$).

The second line contains k pairwise distinct integers v_1, v_2, ..., v_k ($1 \le v_i \le n$) — the indices of the special vertices.

The third line contains n integers c_1, c_2, ..., c_n ($0 \le c_i \le 10^9$).

The fourth line contains m integers w_1, w_2, ..., w_m ($0 \le w_i \le 10^9$).

Then m lines follow, the i-th line contains two integers x_i and y_i ($1 \le x_i, y_i \le n$, $x_i \ne y_i$) — the endpoints of the i-th edge.

There is at most one edge between each pair of vertices.

Output
Print n integers, where the i-th integer is the maximum profit you can get if you have to make the vertex i saturated.

Examples

input
3 2 2 1 3 11 1 5 10 10 1 2 2 3
output
11 2 5

input
4 4 4 1 2 3 4 1 5 7 8 100 100 100 100 1 2 2 3 3 4 1 4
output
21 21 21 21

Note
Consider the first example:

- the best way to make vertex 1 saturated is to direct the edges as $2 \to 1$, $3 \to 2$; 1 is the only saturated vertex, so the answer is 11;
- the best way to make vertex 2 saturated is to leave the edge $1 - 2$ undirected and direct the other edge as $3 \to 2$; 1 and 2 are the saturated vertices, and the cost to leave the edge $1 - 2$ undirected is 10, so the answer is 2;
- the best way to make vertex 3 saturated is to direct the edges as $2 \to 3$, $1 \to 2$; 3 is the only saturated vertex, so the answer is

5.

The best course of action in the second example is to direct the edges along the cycle: $1 \to 2$, $2 \to 3$, $3 \to 4$ and $4 \to 1$. That way, all vertices are saturated.
