Codeforces Round #626 (Div. 2, based on Moscow Open Olympiad in Informatics)

A. Even Subset Sum Problem

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given an array a consisting of n positive integers. Find a **non-empty** subset of its elements such that their sum is **even** (i.e. divisible by 2) or determine that there is no such subset.

Both the given array and required subset may contain equal values.

Input
The first line contains a single integer t ($1 \le t \le 100$), number of test cases to solve. Descriptions of t test cases follow.

A description of each test case consists of two lines. The first line contains a single integer n ($1 \le n \le 100$), length of array a.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 100$), elements of a. The given array a can contain equal values (duplicates).

Output
For each test case output -1 if there is no such subset of elements. Otherwise output positive integer k, number of elements in the required subset. Then output k distinct integers ($1 \le p_i \le n$), indexes of the chosen elements. If there are multiple solutions output any of them.

Example

input
3 3 1 4 3 1 15 2 3 5
output
1 2 -1 2 1 2

Note
There are three test cases in the example.

In the first test case, you can choose the subset consisting of only the second element. Its sum is 4 and it is even.

In the second test case, there is only one non-empty subset of elements consisting of the first element, however sum in it is odd, so there is no solution.

In the third test case, the subset consisting of all array's elements has even sum.

B. Count Subrectangles

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given an array a of length n and array b of length m both consisting of only integers 0 and 1. Consider a matrix c of size $n \times m$ formed by following rule: $c_{i,j} = a_i \cdot b_j$ (i.e. a_i multiplied by b_j). It's easy to see that c consists of only zeroes and ones too.

How many *subrectangles* of size (area) k consisting only of ones are there in c?

A *subrectangle* is an intersection of a consecutive (subsequent) segment of rows and a consecutive (subsequent) segment of columns. I.e. consider four integers x_1, x_2, y_1, y_2 ($1 \le x_1 \le x_2 \le n$, $1 \le y_1 \le y_2 \le m$) a subrectangle $c[x_1 \ldots x_2][y_1 \ldots y_2]$ is an intersection of the rows $x_1, x_1+1, x_1+2, \ldots, x_2$ and the columns $y_1, y_1+1, y_1+2, \ldots, y_2$.

The size (area) of a subrectangle is the total number of cells in it.

Input

The first line contains three integers n, m and k ($1 \le n, m \le 40\,000, 1 \le k \le n \cdot m$), length of array a, length of array b and required size of subrectangles.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 1$), elements of a.

The third line contains m integers $b_1, b_2, \ldots, b_m$ ($0 \le b_i \le 1$), elements of b.

Output
Output single integer — the number of subrectangles of c with size (area) k consisting only of ones.

Examples

input
3 3 2 1 0 1 1 1 1
output
4

input
3 5 4 1 1 1 1 1 1 1 1
output
14

Note
In first example matrix c is:

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

There are 4 subrectangles of size 2 consisting of only ones in it:

$$\begin{pmatrix} \boxed{1} & \boxed{1} & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & \boxed{1} & \boxed{1} \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ \boxed{1} & \boxed{1} & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & \boxed{1} & \boxed{1} \end{pmatrix}$$

In second example matrix c is:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

C. Unusual Competitions

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

A bracketed sequence is called correct (regular) if by inserting "+" and "1" you can get a well-formed mathematical expression from it. For example, sequences "(( )) ( )", "( )" and "(( ) (( )))" are correct, while ") (", "(( )" and "(( ))) (" are not.

The teacher gave Dmitry's class a very strange task — she asked every student to come up with a sequence of arbitrary length, consisting only of opening and closing brackets. After that all the students took turns naming the sequences they had invented. When Dima's turn came, he suddenly realized that all his classmates got the correct bracketed sequence, and whether he got the correct bracketed sequence, he did not know.

Dima suspects now that he simply missed the word "correct" in the task statement, so now he wants to save the situation by modifying his sequence slightly. More precisely, he can **the arbitrary number of times** (possibly zero) perform the *reorder* operation.

The reorder operation consists of choosing an arbitrary consecutive subsegment (substring) of the sequence and then reordering all the characters in it in an arbitrary way. Such operation takes l nanoseconds, where l is the length of the subsegment being reordered. It's easy to see that reorder operation doesn't change the number of opening and closing brackets. For example for ")) ((" he can choose the substring ") (" and do reorder ") ( ) (" (this operation will take 2 nanoseconds).

Since Dima will soon have to answer, he wants to make his sequence correct as fast as possible. Help him to do this, or determine that it's impossible.

Input
The first line contains a single integer n ($1 \le n \le 10^6$) — the length of Dima's sequence.

The second line contains string of length n, consisting of characters "(" and ")" only.

Output
Print a single integer — the minimum number of nanoseconds to make the sequence correct or "-1" if it is impossible to do so.

Examples

input
8))((())(
output
6

input
3 (()
output
-1

Note
In the first example we can firstly reorder the segment from first to the fourth character, replacing it with "()()", the whole sequence will be "()()())(". And then reorder the segment from the seventh to eighth character, replacing it with "()". In the end the sequence will be "()()()()", while the total time spent is $4 + 2 = 6$ nanoseconds.

D. Present

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Catherine received an array of integers as a gift for March 8. Eventually she grew bored with it, and she started calculated various useless characteristics for it. She succeeded to do it for each one she came up with. But when she came up with another one — xor of all pairwise sums of elements in the array, she realized that she couldn't compute it for a very large array, thus she asked for your help. Can you do it? Formally, you need to compute

$$(a_1 + a_2) \oplus (a_1 + a_3) \oplus \ldots \oplus (a_1 + a_n)$$
$$\oplus (a_2 + a_3) \oplus \ldots \oplus (a_2 + a_n)$$
$$\ldots$$
$$\oplus (a_{n-1} + a_n)$$

Here $x \oplus y$ is a bitwise XOR operation (i.e. x ^ y in many modern programming languages). You can read about it in Wikipedia: https://en.wikipedia.org/wiki/Exclusive_or#Bitwise_operation.

Input
The first line contains a single integer n ($2 \le n \le 400\,000$) — the number of integers in the array.

The second line contains integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^7$).

Output
Print a single integer — xor of all pairwise sums of integers in the given array.

Examples

input
2 1 2
output
3

input
3 1 2 3
output
2

Note
In the first sample case there is only one sum $1 + 2 = 3$.

In the second sample case there are three sums: $1 + 2 = 3$, $1 + 3 = 4$, $2 + 3 = 5$. In binary they are represented as $011_2 \oplus 100_2 \oplus 101_2 = 010_2$, thus the answer is 2.

$\oplus$ is the bitwise xor operation. To define $x \oplus y$, consider binary representations of integers x and y. We put the i-th bit of the result to be 1 when exactly one of the i-th bits of x and y is 1. Otherwise, the i-th bit of the result is put to be 0. For example, $0101_2 \oplus 0011_2 = 0110_2$.

E. Instant Noodles

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Wu got hungry after an intense training session, and came to a nearby store to buy his favourite instant noodles. After Wu paid for his purchase, the cashier gave him an interesting task.

You are given a bipartite graph with positive integers in all vertices of the **right** half. For a subset S of vertices of the **left** half we define $N(S)$ as the set of all vertices of the right half adjacent to at least one vertex in S, and $f(S)$ as the sum of all numbers in vertices of $N(S)$. Find the greatest common divisor of $f(S)$ for all possible non-empty subsets S (assume that GCD of empty set is 0).

Wu is too tired after his training to solve this problem. Help him!

Input
The first line contains a single integer t ($1 \le t \le 500\,000$) — the number of test cases in the given test set. Test case descriptions follow.

The first line of each case description contains two integers n and m ($1 \le n, m \le 500\,000$) — the number of vertices in either half of the graph, and the number of edges respectively.

The second line contains n integers c_i ($1 \le c_i \le 10^{12}$). The i-th number describes the integer in the vertex i of the right half of the graph.

Each of the following m lines contains a pair of integers u_i and v_i ($1 \le u_i, v_i \le n$), describing an edge between the vertex u_i of the left half and the vertex v_i of the right half. It is guaranteed that the graph does not contain multiple edges.

Test case descriptions are separated with empty lines. The total value of n across all test cases does not exceed $500\,000$, and the total value of m across all test cases does not exceed $500\,000$ as well.

Output
For each test case print a single integer — the required greatest common divisor.

Example

input
3 2 4 1 1 1 1 1 2 2 1 2 2 3 4 1 1 1 1 1 1 2 2 2 2 3 4 7 36 31 96 29 1 2 1 3 1 4 2 2 2 4 3 1 4 3
output
2 1 12

Note
The greatest common divisor of a set of integers is the largest integer g such that all elements of the set are divisible by g.

In the first sample case vertices of the left half and vertices of the right half are pairwise connected, and $f(S)$ for any non-empty subset is 2, thus the greatest common divisor of these values if also equal to 2.

In the second sample case the subset $\{1\}$ in the left half is connected to vertices $\{1, 2\}$ of the right half, with the sum of numbers equal to 2, and the subset $\{1, 2\}$ in the left half is connected to vertices $\{1, 2, 3\}$ of the right half, with the sum of numbers equal to 3. Thus, $f(\{1\}) = 2$, $f(\{1, 2\}) = 3$, which means that the greatest common divisor of all values of $f(S)$ is 1.

F. Reality Show

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

A popular reality show is recruiting a new cast for the third season! n candidates numbered from 1 to n have been interviewed. The candidate i has aggressiveness level l_i, and recruiting this candidate will cost the show s_i roubles.

The show host reviewes applications of all candidates from $i = 1$ to $i = n$ by increasing of their indices, and for each of them she decides whether to recruit this candidate or not. If aggressiveness level of the candidate i is strictly higher than that of any **already accepted** candidates, then the candidate i will definitely be rejected. Otherwise the host may accept or reject this candidate at her own discretion. The host wants to choose the cast so that to maximize the total *profit*.

The show makes revenue as follows. For each aggressiveness level v a corresponding profitability value c_v is specified, which can be positive as well as negative. All recruited participants enter the stage one by one by increasing of their indices. When the participant i enters the stage, events proceed as follows:

- The show makes c_{l_i} roubles, where l_i is initial aggressiveness level of the participant i.
- If there are two participants with the same aggressiveness level on stage, they immediately start a fight. The outcome of this is:
  - the defeated participant is hospitalized and leaves the show.
  - aggressiveness level of the victorious participant is increased by one, and the show makes c_t roubles, where t is the new aggressiveness level.
- The fights continue until all participants on stage have distinct aggressiveness levels.

It is allowed to select an empty set of participants (to choose neither of the candidates).

The host wants to recruit the cast so that the total profit is maximized. The profit is calculated as the total revenue from the events on stage, less the total expenses to recruit all accepted participants (that is, their total s_i). Help the host to make the show as profitable as possible.

Input

The first line contains two integers n and m ($1 \le n, m \le 2000$) — the number of candidates and an upper bound for initial aggressiveness levels.

The second line contains n integers l_i ($1 \le l_i \le m$) — initial aggressiveness levels of all candidates.

The third line contains n integers s_i ($0 \le s_i \le 5000$) — the costs (in roubles) to recruit each of the candidates.

The fourth line contains $n + m$ integers c_i ($|c_i| \le 5000$) — profitability for each aggrressiveness level.

It is guaranteed that aggressiveness level of any participant can never exceed $n + m$ under given conditions.

Output

Print a single integer — the largest profit of the show.

Examples

input
5 4 4 3 1 2 1 1 2 1 2 1 1 2 3 4 5 6 7 8 9
output
6

input
2 2 1 2 0 0 2 1 -100 -100
output
2

input
5 4

4 3 2 1 1 0 2 6 7 4 12 12 12 6 -3 -5 3 10 -4
output
62

Note

In the first sample case it is optimal to recruit candidates $1, 2, 3, 5$. Then the show will pay $1 + 2 + 1 + 1 = 5$ roubles for recruitment. The events on stage will proceed as follows:

- a participant with aggressiveness level 4 enters the stage, the show makes 4 roubles;
- a participant with aggressiveness level 3 enters the stage, the show makes 3 roubles;
- a participant with aggressiveness level 1 enters the stage, the show makes 1 rouble;
- a participant with aggressiveness level 1 enters the stage, the show makes 1 roubles, a fight starts. One of the participants leaves, the other one increases his aggressiveness level to 2. The show will make extra 2 roubles for this.

Total revenue of the show will be $4 + 3 + 1 + 1 + 2 = 11$ roubles, and the profit is $11 - 5 = 6$ roubles.

In the second sample case it is impossible to recruit both candidates since the second one has higher aggressiveness, thus it is better to recruit the candidate 1.