Problem A. Find Values

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.1 second
Memory limit:	256 mebibytes

It is well known that each team participating in the *MG Cup* consists of 4 members. Let's denote their total scores as s_1, s_2, s_3, and s_4 respectively.

This year, the organizers decided to make the contest a bit more interesting. Therefore, instead of showing the exact scores to each contestant, they decided to only show the sum of points achieved by all the other contestants from the same team. I.e. instead of s_1 they showed $s_2 + s_3 + s_4$, instead of s_2, the value $s_1 + s_3 + s_4$, instead of s_3, the value $s_1 + s_2 + s_4$ and instead of s_4 the value $s_1 + s_2 + s_3$.

Help the team retrieve their original scores. It is guaranteed that the solution exists and is unique.

Input

The first line of input contains four integers $s_2 + s_3 + s_4$, $s_1 + s_3 + s_4$, $s_1 + s_2 + s_4$, and $s_1 + s_2 + s_3$.

Output

On the first line, print the four integers s_1, s_2, s_3, and s_4. It is guaranteed that the solution exists and is unique. **It is guaranteed that s_1, s_2, s_3, and s_4 are integers.**

Example

standard input	standard output
21 16 24 23	7 12 4 5

Explanation

Let $s_1 = 7$, $s_2 = 12$, $s_3 = 4$ and $s_4 = 5$. Then:

- $s_2 + s_3 + s_4 = 21$
- $s_1 + s_3 + s_4 = 16$
- $s_1 + s_2 + s_4 = 24$
- $s_1 + s_2 + s_3 = 23$

Constraints

- $0 \leq s_1, s_2, s_3, s_4 \leq 10^6$.

Scoring

This problem has 100 test cases (excluding the example given above). Each test case is worth one point and is scored separately. Moreover, it is guaranteed that:

- In 20 test cases $s_1 = s_2 = s_3 = s_4$.
- In additional 20 test cases $0 \leq s_1, s_2, s_3, s_4 \leq 1$.
- In the remaining 60 test cases there are no additional constraints.

Problem B. Split Array

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

Given an array with n positive integers. Find the number of ways to split it into an arbitrary number of consecutive segments such that the sum of each segment is exactly two times the sum of the previous one. Notice that there is always at least one way to do this (i.e. to select a single segment equal to the entire original array).

Input

The first line of input contains one integer, n, representing the size of the array. The second line contains n integers, representing the elements of the array.

Output

On the first line, print the single integer, the number of ways to split the original array into consecutive segments such that the sum of each segment is exactly two times greater than the sum of the previous segment.

Example

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

Explanation

Consider the following three divisions:

- $[5, 1, 8, 3, 1, 6, 7, 9, 2]$ - one segment with sum 42.
- $[5, 1, 8], [3, 1, 6, 7, 9, 2]$ - two segments with sums 14 and 28 respectively.
- $[5, 1], [8, 3, 1], [6, 7, 9, 2]$ - three segments with sums 6, 12 and 24 respectively.

Constraints

- $1 \leq n \leq 10^6$.
- $1 \leq a_i \leq 10^9$.

Scoring

This problem has 6 subtasks. You get the points for each subtask if and only if your solutions evaluate correctly on all test cases in that subtask:

- Subtask 1 [6 points]: $n \leq 20$, $a_i = 1$.
- Subtask 2 [11 points]: $n \leq 20$.
- Subtask 3 [13 points]: $n \leq 200$.
- Subtask 4 [18 points]: $n \leq 1000$.

- Subtask 5 [19 points]: $a_i = 1$.
- Subtask 6 [33 points]: No additional constraints.

Notice that you will see subtask 0 when submitting the problem. This subtask is worth 0 points and contains only one test case, i.e. the test cases described in the statement above.

Problem C. Missing Lengths

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 seconds
Memory limit:	256 mebibytes

The ancient country of Aibres had n cities and m bidirectional roads between them. The capital of this country, Edargleb was the city indexed with the number 1. It is known that any other city could have been reached starting from Edargleb. Roads of Aibres had varying lengths and most of them are known. Unfortunately, some of those lengths are lost to the sands of time! However, it is known that **all of the missing lengths are equal**.

Through extensive archeological excavations, the record listing the distances from Edargleb to all of the other cities was also discovered.

Based on this information, try to find the missing lengths. It is guaranteed that the solution with road length in interval $[1, 10^9]$ exists. **If there are multiple solutions, output the smallest one.**

Input

The first line contains two integers n and m, representing the number of cities and the number of roads respectively. The following m lines contain three integers each, u, v, and w. These lines describe a bidirectional road between cities u and v with length w. If the length of a road isn't known, the value w is equal to -1.

The last line contains n integers, the shortest distances from city 1 to all the other cities.

Output

On the first line, print the single integer, the unknown length. It is guaranteed that the solution exists. If there are multiple solutions, output the smallest one.

Example

standard input	standard output
4 4 1 2 3 1 3 4 2 4 -1 3 4 -1 0 3 4 5	2

Explanation

Let the missing length be 2. Then, the shortest paths are:

- City 1 to itself, with zero roads on the path and total length 0.
- From city 1 to city 2, path $1 - 2$, with total length 3.
- From city 1 to city 3, path $1 - 3$, with total length 4.
- From city 1 to city 4, path $1 - 2 - 4$, with total length 5.

Constraints

- $1 \leq n \leq 200.000$.
- $1 \leq m \leq 500.000$.
- For each road $u \neq v$.
- It is guaranteed that you can reach any other city starting from Edargleb.
- All the lengths (including the missing one) are positive integers up to 10^9.

Scoring

This problem has 5 subtasks. You get the points for each subtask if and only if your solutions evaluate correctly on all test cases in that subtask:

- Subtask 1 [7 points]: $m = n - 1$ and $|v - u| = 1$ for each road.
- Subtask 2 [14 points]: $n, m \leq 1000$, and the missing length is in interval $[1, 1000]$.
- Subtask 3 [17 points]: $n, m \leq 2000$.
- Subtask 4 [21 points]: $n \leq 50.000$, $m \leq 200.000$
- Subtask 5 [41 points]: No additional constraints.

Notice that you will see subtask 0 when submitting the problem. This subtask is worth 0 points and contains only one test case, i.e. the test cases described in the statement above.

Problem D. Count Intervals

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	256 mebibytes

Given an array a with n non-negative integers. Find the number of non-empty intervals $[l, r]$ such that:

- $[l, r]$ isn't equal to the entire array and
- a_l OR a_{l+1} OR ... OR a_{r-1} OR $a_r = a_1$ OR ... OR a_{l-1} OR a_{r+1} OR ... OR a_n

Where OR denotes the bitwise or operation (see the notes section for more details).

Input

The first line of input contains one integer, n, representing the size of the array. The second line contains n integers, representing the elements of the array.

Output

On the first line, print the single integer, the number of ways to choose interval $[l, r]$.

Example

standard input	standard output
5 1 3 1 2 2	6

Explanation

Valid choices are $[1, 2], [3, 4], [2, 3], [3, 5], [2, 4], [2, 2]$.

Constraints

- $1 \leq n \leq 10^6$.
- $0 \leq a_i \leq 10^{18}$.

Notes

Operator OR represents bitwise or operation.

Bitwise or operator is denoted with 'or' in Pascal, and with '|' in C++. The operation x OR y is defined for non-negative integers x, y in the following way: we first write the numbers in base two (i.e. in binary format). If one of the numbers has fewer digits than the other one, we prefix it with leading zeroes, until both numbers have the same number of digits. In such a way, we can observe two arrays of binary digits. Let's denote them with $a_1, \ldots, a_k$ and $b_1, \ldots b_k$. Then for each position $i \in \{1, \ldots, k\}$ we calculate c_i using the following set of rules:

- For $a_i = 0, b_i = 0$ we write $c_i = 0$
- For $a_i = 0, b_i = 1$ we write $c_i = 1$
- For $a_i = 1, b_i = 0$ we write $c_i = 1$
- For $a_i = 1, b_i = 1$ we write $c_i = 1$

An array of binary digits $c_1, \ldots, c_k$ represents the binary format of the result, x OR y.

Scoring

This problem has 5 subtasks. You get the points for each subtask if and only if your solutions evaluate correctly on all test cases in that subtask:

- Subtask 1 [9 points]: $n \leq 100$
- Subtask 2 [14 points]: $n \leq 5000$.
- Subtask 3 [24 points]: $n \leq 10^5$ and $a_i \leq 1$.
- Subtask 4 [37 points]: $n \leq 10^5$.
- Subtask 5 [16 points]: No additional constraints.

Notice that you will see subtask 0 when submitting the problem. This subtask is worth 0 points and contains only one test case, i.e. the test cases described in the statement above.