2. Subarray Sums

Subarray sums are often useful in finding the cumulative frequency of a given interval. In this problem, your goal is to find the subarray sums of the given array for the given queries.

You are given a 1-indexed array, *numbers*, of length *n*. The number of queries is given to you as *q*. Each query is defined by three integers: start index *l*, end index *r*, and a number *x*. For each query, find the sum of numbers between indexes *l* and *r* (both extremes included), and for each occurrence of zero within the range, add the value of *x* to the sum.

For example, let's say there are *n = 4* numbers, where the array *numbers = [20, 30, 0, 10]*. Also, there is *q = 1* query, the **start index is *l* = *1*, and the end index is *r* = *3*. For each occurrence of zero within the range, we'll add *x = 10 to it*. So, for this example, we are looking for the sum of the numbers between index 1 and index 3, which gives us 20 + 30 + 0 = 50. Because there is 1 zero in this range, we also add 10 to it, so the final answer is 50 + 10 = 60.**

Function Description
Complete the function *findSum* in the editor below.

findSum has the following parameter(s):
  int *numbers[n]*: the array of integers, 1-indexed, that will be queried
  int *queries[q][3]*: a 2-dimensional array of integers, 0-indexed, containing **start index *l*, end index *r*, and *x* for each query**
Returns:
  long int *arr[q]*: the subarray sums of the given queries

Constraints

- $1 \le n \le 10^5$
- $1 \le q \le 10^5$
- $-10^9 \le numbers[i] \le 10^9$
- length of *queries[i]* = 3, for all *i*
- $1 \le queries[i][0] \le queries[i][1] \le n$
- $-10^9 \le queries[i][2] \le 10^9$

▼ Input Format for Custom Testing

The first line contains an integer, *n*, denoting the length of the array *numbers*.
Each line *i* of the *n* subsequent lines (where *1 ≤ i ≤ n*) contains an integer that describes *numbers[i]*.
Then the next line contains an integer, *q*, denoting the number of queries.
The next line contains an integer, 3, denoting the number of integers in each query.
Each line *i* of the *q* subsequent lines (where *0 ≤ i < q*) contains three space-separated integers—**start index *l*, end index *r*, and *x*—for *queries[i]*.**

▼ Sample Case 0

```
3
5
10
10
1
3
1 2 5
```

Sample Output

```
15
```

Explanation
Here, *numbers = [5, 10, 10]*. There is a single query for the range [1, 2]. The sum of the numbers in this range is 15. As there are no zeroes in the queried range, the answer remains 15.

▼ Sample Case 1

```
2
-5
0
2
3
2 2 20
1 2 10
```

Sample Output

```
20
5
```

Explanation

Here, *numbers = [-5, 0]*. There are 2 queries. The first query is for the range [2, 2], which equals a sum of 0. Because there is one zero in the queried range, *x = 20* is added to 0 to get the final answer of 20. The second query is for the range [1, 2], which equals a sum of -5. Because there is one zero in this queried range, *x = 10* is added to -5 to get the final answer of 5.