Q:-1

**Minimum In SubArray**

Send Feedback

**Range Minimum Query**

Given an array A of size N, there are two types of queries on this array.

1) q l r: In this query you need to print the minimum in the sub-array A[l:r].

**2) u x y: In this query you need to update A[x]=y.**

**Input:**

First line of the test case contains two integers, N and Q, size of array A and number of queries.

Second line contains N space separated integers, elements of A.

Next Q lines contain one of the two queries.

**Output:**

For each type 1 query, print the minimum element in the sub-array A[l:r].

**Contraints:**

1≤N,Q,y≤10^5

1≤l,r,x≤N

**Sample Input :**

5 5

1 5 2 4 3

q 1 5

q 1 3

q 3 5

u 3 6

q 1 5

**Sample Output :**

1

1

2

1

Q2:-

**Maximum Pair Sum**

Send Feedback

You are given a sequence A[1], A[2], ..., A[N], ( 0 ≤ A[i] ≤ 10^8 , 2 ≤ N ≤ 10^5 ). There are two types of operations and they are defined as follows:

**Update:**

This will be indicated in the input by a 'U' followed by space and then two integers i and x.

U i x, 1 ≤ i ≤ N, and x, 0 ≤ x ≤ 10^8.

This operation sets the value of A[i] to x.

**Query:**

This will be indicated in the input by a 'Q' followed by a single space and then two integers i and j.

Q x y, 1 ≤ x < y ≤ N.

You must find i and j such that x ≤ i, j ≤ y and i != j, such that the sum A[i]+A[j] is maximized. Print the sum A[i]+A[j].

**Input**

The first line of input consists of an integer N representing the length of the sequence.

Next line consists of N space separated integers A[i]. Next line contains an integer Q, Q ≤ 10^5, representing the number of operations. Next Q lines contain the operations.

**Output**

Output the maximum sum mentioned above, in a separate line, for each Query.

**Input:**

5

1 2 3 4 5

6

Q 2 4

Q 2 5

U 1 6

Q 1 5

U 1 7

Q 1 5

**Output:**

7

9

11

12

Q3:-

**Maximum Sum In Subarray**

Send Feedback

You are given a sequence A[1], A[2], ..., A[N] . ( |A[i]| ≤ 15007 , 1 ≤ N ≤ 50000 ). A query is defined as follows:

Query(x,y) = Max { a[i]+a[i+1]+...+a[j] ; x ≤ i ≤ j ≤ y }.

Given M queries, your program must output the results of these queries.

**Input**

The first line of the input file contains the integer N.

In the second line, N numbers follow.

The third line contains the integer M.

M lines follow, where line i contains 2 numbers xi and yi.

**Output**

Your program should output the results of the M queries, one

query per line.

**Sample Input:**

3

-1 2 3

1

1 2

**Sample Output:**

2

Q:4

**Sum Of Squares**

Send Feedback

Segment trees are extremely useful. In particular "Lazy Propagation" (i.e. see here, for example) allows one to compute sums over a range in O(lg(n)), and update ranges in O(lg(n)) as well. In this problem you will compute something much harder:

The sum of squares over a range with range updates of 2 types:

1) increment in a range

2) set all numbers the same in a range.

**Input**

There will be T (T <= 25) test cases in the input file. First line of the input contains two positive integers, N (N <= 100,000) and Q (Q <= 100,000).

The next line contains N integers, each at most 1000. Each of the next Q lines starts with a number, which indicates the type of operation:

2 st nd -- return the sum of the squares of the numbers with indices in [st, nd] {i.e., from st to nd inclusive} (1 <= st <= nd <= N).

1 st nd x -- add "x" to all numbers with indices in [st, nd] (1 <= st <= nd <= N, and -1,000 <= x <= 1,000).

0 st nd x -- set all numbers with indices in [st, nd] to "x" (1 <= st <= nd <= N, and -1,000 <= x <= 1,000).

Output

For each test case output the “Case <caseno>:” in the first line and from the second line output the sum of squares for each operation of type 2. Intermediate overflow will not occur with proper use of 64-bit signed integer.

**Input:**

2

4 5

1 2 3 4

2 1 4

0 3 4 1

2 1 4

1 3 4 1

2 1 4

1 1

1

2 1 1

**Output:**

Case 1:

30

7

13

Case 2:

1

**Q:-5**

**Counting Even/Odd**

Send Feedback

Ashu and Shanu are best buddies. One day Shanu gives Ashu a problem to test his intelligence.He gives him an array of N natural numbers and asks him to solve the following queries:-

Query 0 :- modify the element present at index i to x.

Query 1:- count the number of even numbers in range l to r inclusive.

Query 2:- count the number of odd numbers in range l to r inclusive.

**Input:**

First line of the input contains the number N. Next line contains N natural numbers.

Next line contains an integer Q followed by Q queries.

0 x y - modify the number at index x to y.

1 x y - count the number of even numbers in range l to r inclusive.

2 x y - count the number of odd numbers in range l to r inclusive.

**Constraints:**

1<=N,Q<=10^5

1<=l<=r<=N

0<=Ai<=10^9

1<=x<=N

0<=y<=10^9

**Note:-**

indexing starts from 1.

**Sample Input**

6

1 2 3 4 5 6

4

1 2 5

2 1 4

0 5 4

1 1 6

**Sample Output**

2

2

4