Kick Start 2020 - Round C

Candies

Problem

Carl has an array of **N** candies. The i-th element of the array (indexed starting from 1) is $\mathbf{A_i}$ representing *sweetness value* of the i-th candy. He would like to perform a series of **Q** operations. There are two types of operation:

- Update the sweetness value of a candy in the array.
- Query the *sweetness score* of a subarray.

The sweetness score of a subarray from index l to r is: $\mathbf{A_l} \times 1 - \mathbf{A_{l+1}} \times 2 + \mathbf{A_{l+2}} \times 3 - \mathbf{A_{l+3}} \times 4 + \mathbf{A_{l+4}} \times 5 \ldots$

More formally, the sweetness score is the sum of $(-1)^{i-l}\mathbf{A_i} \times (i - l + 1)$, for all i from l to r inclusive.

For example, the sweetness score of:

- [3, 1, 6] is $3 \times 1 - 1 \times 2 + 6 \times 3 = 19$
- [40, 30, 20, 10] is $40 \times 1 - 30 \times 2 + 20 \times 3 - 10 \times 4 = 0$
- [2, 100] is $2 \times 1 - 100 \times 2 = -198$

Carl is interested in finding out the total sum of sweetness scores of all queries. If there is no query operation, the sum is considered to be 0. Can you help Carl find the sum?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case begins with a line containing **N** and **Q**. The second line contains **N** integers describing the array. The i-th integer is $\mathbf{A_i}$. The j-th of the following **Q** lines describe the j-th operation. Each line begins with a single character describing the type of operation (`U` for update, `Q` for query).

- For an update operation, two integers $\mathbf{X_j}$ and $\mathbf{V_j}$ follow, indicating that the $\mathbf{X_j}$-th element of the array is changed to $\mathbf{V_j}$.
- For a query operation, two integers $\mathbf{L_j}$ and $\mathbf{R_j}$ follow, querying the sweetness score of the subarray from the $\mathbf{L_j}$-th element to the $\mathbf{R_j}$-th element (inclusive).

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is the total sum of sweetness scores of all the queries.

Limits

Time limit: 20 seconds.
Memory limit: 1 GB.
$1 \le \mathbf{T} \le 100$.
$1 \le \mathbf{A_i} \le 100$, for all i.
$1 \le \mathbf{N} \le 2 \times 10^5$ and $1 \le \mathbf{Q} \le 10^5$ for at most 6 test cases.
For the remaining cases, $1 \le \mathbf{N} \le 300$ and $1 \le \mathbf{Q} \le 300$.

If the j-th operation is an update operation, $1 \le \mathbf{X_j} \le \mathbf{N}$ and $1 \le \mathbf{V_j} \le 100$.
If the j-th operation is a query operation, $1 \le \mathbf{L_j} \le \mathbf{R_j} \le \mathbf{N}$.

Test Set 1

There will be at most 5 update operations.

Test Set 2

There are no special constraints.

Sample

Sample Input

```
2
5 4
1 3 9 8 2
Q 2 4
Q 5 5
U 2 10
Q 1 2
3 3
4 5 5
U 1 2
U 1 7
Q 1 2
```

Sample Output

```
Case #1: -8
Case #2: -3
```

In sample case #1:

- The first query asks for the sweetness score of [3, 9, 8] which is 3 × 1 - 9 × 2 + 8 × 3 = 9.
- The second query asks for the sweetness score of [2] which is 2 × 1 = 2.
- The third query asks for the sweetness score of [1, 10] which is 1 × 1 - 10 × 2 = -19.

Thus, the final output should be 9 + 2 - 19 = -8.

In sample case #2:

- The first and only query asks for the sweetness score of [7, 5] which is 7 × 1 - 5 × 2 = -3.

Thus, the final output should be -3.