```cpp
#include <iostream>
#include <cstring>
#include <cstdio>

using namespace std;

struct node
{
    int l, r;
    long long sum;
    long long lazy;
}tree[200005];

int a[200005];


void push_up(int i)
{
    tree[i].sum = tree[i * 2].sum + tree[i * 2 + 1].sum;
}

void push_down(int i)
{
    if (tree[i].lazy == 0)
        return;
    tree[i * 2].sum += (tree[i * 2].r - tree[i * 2].l + 1) * tree[i].lazy;
    tree[i * 2].lazy += tree[i].lazy;
    tree[i * 2 + 1].sum += (tree[i * 2 + 1].r - tree[i * 2 + 1].l + 1) * tree[i].lazy;
    tree[i * 2 + 1].lazy += tree[i].lazy;
    tree[i].lazy = 0;
}

void build(int i, int l, int r)
{
    tree[i].l = l;
    tree[i].r = r;
    tree[i].lazy = 0;
    if (l == r)
    {
        tree[i].sum = a[l];
        return;
    }
    int mid = (l + r) / 2;
    build(i * 2, l, mid);
    build(i * 2 + 1, mid + 1, r);
    push_up(i);

}

void update(int i, int l, int r, int val)
{
    if (tree[i].l > r || tree[i].r < l)
        return;
    if (tree[i].r <= r && tree[i].l >= l)
    {
        tree[i].sum += (tree[i].r - tree[i].l + 1) * val;
        tree[i].lazy += val;
        return;
    }
    push_down(i);
    update(i * 2, l, r, val);
    update(i * 2 + 1, l, r, val);
    push_up(i);


}


long long query(int i, int l, int r)
{
    if (tree[i].l > r || tree[i].r < l)
        return 0;
    if (tree[i].l >= l && tree[i].r <= r)
    {
        return tree[i].sum;
    }

    push_down(i);
    return query(i * 2, l, r) + query(i * 2 + 1, l, r);
}


int main()
{
    int t, k = 1;
    cin >> t;
    while (t--)
```

```
{
    cout << "Case " << k++ << ":" << endl;
    int n;
    scanf("%d", &n);
    for (int i = 1; i <= n; i++)
        scanf("%d", &a[i]);
    build(1, 1, n);
    string c;
    while (cin >> c)
    {
        if (c == "End")
            break;
        int i, j;
        cin >> i >> j;
        if (c == "Add")
        {
            update(1, i, i, j);
        }
        else if (c == "Sub")
        {
            update(1, i, i, -1 * j);
        }
        else
        {
            cout << query(1, i, j) << endl;
        }
    }
}
return 0;
}
```