// Дерево отрезков с массовыми операциями. Максимум и присвоение на отрезке

#include <iostream>

#include <vector>

#include <string>

using namespace std;

struct node {

long long val;

long long push;

node(long long val, long long push): val{val}, push{push} {}

};

int n, m;

vector<node> tree;

vector<long long> a;

const long long INF = 1e18;

// Инвариант:

// node.val - АКТУАЛЬНОЕ значение максимального элемента в данной вершине

// push - значение push - величины вершины

ostream& operator<< (ostream& stream, const node& a) {

stream << "{" << a.val << " " << a.push << "}";

return stream;

}

void print() {

cout << "\n" << string(100, '#') << "\n";

for (int i = 0; i < 4 \* n + 1; i++) {

cout << i << ": " << tree[i] << "\n";

}

cout << "\n" << string(100, '#') << "\n";

}

node merge(const node& a, const node& b) {

if (a.val > b.val) {

return node(a.val, 0);

}

return node(b.val, 0);

}

void build(int ver, int l, int r) {

if (r - l == 1) {

tree[ver] = node(a[l], 0);

return;

}

int m = (l + r) / 2;

build(ver \* 2, l, m);

build(ver \* 2 + 1, m, r);

tree[ver] = merge(tree[ver \* 2], tree[ver \* 2 + 1]);

}

void push(int ver) {

long long delta = tree[ver].push;

tree[ver \* 2].val += delta;

tree[ver \* 2].push += delta;

tree[ver \* 2 + 1].val += delta;

tree[ver \* 2 + 1].push += delta;

tree[ver].push = 0;

}

void add(int ver, int l, int r, int ql, int qr, long long x) {

if (l >= qr || r <= ql) {

return;

}

if (ql <= l && r <= qr) {

tree[ver].push += x;

tree[ver].val += x;

return;

}

push(ver);

int m = (l + r) / 2;

add(ver \* 2, l, m, ql, qr, x);

add(ver \* 2 + 1, m, r, ql, qr, x);

tree[ver] = merge(tree[ver \* 2], tree[ver \* 2 + 1]);

}

node get(int ver, int l, int r, int ql, int qr) {

if (l >= qr || r <= ql) {

return node(-INF, 0);

}

if (ql <= l && r <= qr) {

return tree[ver];

}

push(ver);

int m = (l + r) / 2;

return merge(get(ver \* 2, l, m, ql, qr), get(ver \* 2 + 1, m, r, ql, qr));

}

int main() {

cin >> n;

a.resize(n);

tree.resize(4 \* n + 1, node(0, 0));

for (int i = 0; i < n; i++) {

cin >> a[i];

}

build(1, 0, n);

// print();

cin >> m;

for (int i = 0; i < m; i++) {

char t;

cin >> t;

if (t == 'm') {

int l, r;

cin >> l >> r;

cout << get(1, 0, n, l - 1, r).val << " ";

} else {

int l, r;

long long x;

cin >> l >> r >> x;

add(1, 0, n, l - 1, r, x);

// print();

}

}

}