```cpp
const int MAXN = 100010;
int tree[4*MAXN];
int lazy[4*MAXN];
int v[MAXN];

void build(int idx, int l, int r) {
    if(l==r) {
        tree[idx] = v[l];
    }
    else {
        int m = (l+r)/2;
        build(2*idx,l,m);
        build(2*idx+1,m+1,r);
        tree[idx] = tree[2*idx] + tree[2*idx+1];
    }
}

void go(int idx, int l, int r) {
    if(lazy[idx] != 0) {    //needs to be updated
        tree[idx] += (r-l+1)*lazy[idx];
        if(l!=r) { //not leaf
            lazy[2*idx] += lazy[idx];
            lazy[2*idx+1] += lazy[idx];
        }
        lazy[idx] = 0; //reset it
    }
}

void update(int px, int py, int val, int idx, int l, int r) {
    //it needs to come first because of the lines 64 to 66
    go(idx,l,r);
    if(py<l || px>r) return;
    if(l>=px && r<=py) {    //fully within range
        tree[idx] += (r-l+1)*val;
        if(l!=r) { //not leaf
            lazy[2*idx] += val;
            lazy[2*idx+1] += val;
        }
        return;
    }
    int m = (l+r)/2;
    update(px,py,val,2*idx,l,m);
    update(px,py,val,2*idx+1,m+1,r);
    tree[idx] = tree[2*idx] + tree[2*idx+1];
}

int query(int px, int py, int idx, int l, int r) {
    if(py<l || px>r) {
        return 0;
    }
    go(idx,l,r);
    if(l>=px && r<=py) {
        return tree[idx];
    }
    int m = (l+r)/2;
    int p1 = query(px,py,2*idx,l,m);
    int p2 = query(px,py,2*idx+1,m+1,r);
    return p1+p2;
}
```