```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;

struct treeNode {
    ll sqSum;
    ll tSum;
};

struct lazyNode {
    ll set;
    ll update;
};

void makeTree(int s, int e, int i, treeNode* tree, ll* arr) {
    if(s == e) {
        tree[i].sqSum = arr[s - 1]*arr[s - 1];
        tree[i].tSum = arr[s - 1];
        return;
    }
    int mid = (s + e)/2;
    makeTree(s, mid, 2*i, tree, arr);
    makeTree(mid + 1, e, 2*i + 1, tree, arr);
    tree[i].sqSum = tree[2*i].sqSum + tree[2*i + 1].sqSum;
    tree[i].tSum = tree[2*i].tSum + tree[2*i + 1].tSum;
}

void update(int s, int e, int l, int r, ll v, int qtype, int i, treeNode* tree, lazyNode*
lazyTree) {
    if(s > e) return;
    if(lazyTree[i].set != 0) {
        ll x = lazyTree[i].set;
        tree[i].tSum = x*(e - s + 1);
        tree[i].sqSum = x*x*(e - s + 1);
        if(s != e) {
            lazyTree[2*i].update = 0;
            lazyTree[2*i].set = x;
            lazyTree[2*i + 1].update = 0;
            lazyTree[2*i + 1].set = x;
        }
        lazyTree[i].set = 0;
    }
    if(lazyTree[i].update != 0) {
        ll x = lazyTree[i].update;
        tree[i].sqSum += x*x*(e - s + 1) + 2*x*(tree[i].tSum);
        tree[i].tSum += x*(e - s + 1);
        if(s != e) {
            lazyTree[2*i].update += x;
            lazyTree[2*i + 1].update += x;
        }
        lazyTree[i].update = 0;
    }
    if(e < l || s > r) {
        return;
    }
    if(s >= l && e <= r) {
        if(qtype == 0) {
            tree[i].tSum = v*(e - s + 1);
            tree[i].sqSum = v*v*(e - s + 1);
            if(s != e) {
                lazyTree[2*i].update = 0;
                lazyTree[2*i].set = v;
                lazyTree[2*i + 1].update = 0;
                lazyTree[2*i + 1].set = v;
```

```
            }
        } else {
            tree[i].sqSum += v*v*(e - s + 1) + 2*v*(tree[i].tSum);
            tree[i].tSum += v*(e - s + 1);
            if(s != e) {
                lazyTree[2*i].update += v;
                lazyTree[2*i + 1].update += v;
            }
        }
        return;
    }
    int mid = (s + e)/2;
    update(s, mid, l, r, v, qtype, 2*i, tree, lazyTree);
    update(mid + 1, e, l, r, v, qtype, 2*i + 1, tree, lazyTree);
    tree[i].tSum = tree[2*i].tSum + tree[2*i].tSum;
    tree[i].sqSum = tree[2*i].sqSum + tree[2*i + 1].sqSum;
}

ll query(int s, int e, int l, int r, int i, treeNode* tree, lazyNode* lazyTree) {
    if(s > e) return 0;
    if(lazyTree[i].set != 0) {
        ll x = lazyTree[i].set;
        tree[i].tSum = x*(e - s + 1);
        tree[i].sqSum = x*x*(e - s + 1);
        if(s != e) {
            lazyTree[2*i].update = 0;
            lazyTree[2*i].set = x;
            lazyTree[2*i + 1].update = 0;
            lazyTree[2*i + 1].set = x;
        }
        lazyTree[i].set = 0;
    }
    if(lazyTree[i].update != 0) {
        ll x = lazyTree[i].update;
        tree[i].sqSum += x*x*(e - s + 1) + 2*x*(tree[i].tSum);
        tree[i].tSum += x*(e - s + 1);
        if(s != e) {
            lazyTree[2*i].update += x;
            lazyTree[2*i + 1].update += x;
        }
        lazyTree[i].update = 0;
    }
    if(e < l || s > r) {
        return 0;
    }
    if(s >= l && e <= r) {
        return tree[i].sqSum;
    }
    int mid = (s + e)/2;
    ll a1 = query(s, mid, l, r, 2*i, tree, lazyTree);
    ll a2 = query(mid + 1, e, l, r, 2*i + 1, tree, lazyTree);
    return a1 + a2;
}

int main() {
    int test_n;
    cin >> test_n;
    for(int i = 0 ; i < test_n; i++) {
        cout << "Case " << i + 1 << ":" << endl;
        int n, q;
        cin >> n >> q;
        ll arr[n];
        for(int j = 0; j < n; j++) cin >> arr[j];
        treeNode tree[4*n];
        makeTree(1, n, 1, tree, arr);
```

```
        lazyNode lazyTree[4*n];
        for(int j = 0 ; j < 4*n; j++) {
            lazyTree[j].set = 0;
            lazyTree[j].update = 0;
        }
        for(int j = 0; j < q; j++) {
            int qtype;
            cin >> qtype;
            if(qtype == 0 || qtype == 1) {
                int l, r;
                ll v;
                cin >> l >> r >> v;
                update(1, n, l, r, v, qtype, 1, tree, lazyTree);
            } else if(qtype == 2) {
                int l, r;
                cin >> l >> r;
                cout << query(1, n, l, r, 1, tree, lazyTree) << endl;
            }
        }
    }
```