```
#include <algorithm>
#include <cstring>
#include <iostream>
using namespace std;

#define N 100005
#define LL long long
#define lc p << 1
#define rc p << 1 | 1
struct Tree { // 线段树
    LL l, r, sum, add;
} tr[N * 4];
LL n, w[N];

void pushup(LL p)
{
    tr[p].sum = tr[lc].sum + tr[rc].sum;
}
void pushdown(LL p)
{
    auto &u = tr[p], &l = tr[lc], &r = tr[rc];
    if (u.add) {
        l.sum += u.add * (l.r - l.l + 1),
            r.sum += u.add * (r.r - r.l + 1),
            l.add += u.add,
            r.add += u.add,
            u.add = 0;
    }
}
void build(LL p, LL l, LL r)
{
    tr[p] = { l, r, w[l], 0 }; // 赋值
    if (l == r)
        return; // 叶子
    LL m = l + r >> 1; // 裂开
    build(lc, l, m);
    build(rc, m + 1, r);
    pushup(p);
}
void update(LL p, LL x, LL y, LL k)
{
    if (x > tr[p].r || y < tr[p].l)
        return; // 越界
    if (x <= tr[p].l && tr[p].r <= y) { // 覆盖
        tr[p].sum += (tr[p].r - tr[p].l + 1) * k;
        tr[p].add += k;
        return;
    }
    pushdown(p); // 裂开
    update(lc, x, y, k);
    update(rc, x, y, k);
    pushup(p);
}
LL query(LL p, LL x, LL y)
{
    if (x > tr[p].r || y < tr[p].l)
        return 0; // 越界
    if (x <= tr[p].l && tr[p].r <= y) // 覆盖
        return tr[p].sum;

    pushdown(p); // 裂开
    LL sum = 0;
    sum += query(lc, x, y) + query(rc, x, y);
    return sum;
}
int main()
{
    LL m, op, x, y, k;
    cin >> n >> m;
    for (LL i = 1; i <= n; i++)
        cin >> w[i];
    build(1, 1, n);

    while (m--) {
        cin >> op >> x >> y;
        if (op == 2)
            cout << query(1, x, y) << endl;
        else
            cin >> k, update(1, x, y, k);
    }
    return 0;
}
```