#include <bits/stdc++.h>

using namespace std;

const int MAXN = 100005;

typedef long long ll;

int n, m;

// 栈数组大小一般最大为1MB到8MB,1MB最大为26万

// 线段树节点总数理论不超过2n,一般开4n的空间

ll tree[MAXN << 2], mark[MAXN << 2], a[MAXN];

// cl,cr对应节点的区间

void build(int p = 1, int cl = 1, int cr = n) // 建树,时间复杂度为o(N)

{

    if (cl == cr)

        return void(tree[p] = a[cl]);

    int mid = (cl + cr) >> 1;

    build(p << 1, cl, mid);

    build(p << 1 | 1, mid + 1, cr);

    tree[p] = tree[p << 1] + tree[p << 1 | 1];

}

void push\_down(int p, int len) // 懒标记下推

{

    if (len <= 1)

        return;

    tree[p << 1] += mark[p] \* (len - len / 2);

    mark[p << 1] += mark[p];

    tree[p << 1 | 1] += mark[p] \* (len / 2);

    mark[p << 1 | 1] += mark[p];

    mark[p] = 0;

}

// l和r代表该节点代表的范围

ll query(int l, int r, int p = 1, int cl = 1, int cr = n) // 区间查询

{

    if (cl >= l && cr <= r)

        return tree[p];

    push\_down(p, cr - cl + 1);

    ll mid = (cl + cr) >> 1, ans = 0;

    if (mid >= l)

        ans += query(l, r, p << 1, cl, mid);

    if (mid < r)

        ans += query(l, r, p << 1 | 1, mid + 1, cr);

    return ans;

}

void update(int l, int r, int d, int p = 1, int cl = 1, int cr = n)

{

    if (cl >= l && cr <= r)

        return void((tree[p] += d \* (cr - cl + 1), mark[p] += d));

     push\_down(p, cr - cl + 1);

    int mid = (cl + cr) >> 1;

    if (mid >= l)

        update(l, r, d, p << 1, cl, mid);

    if (mid < r)

        update(l, r, d, p << 1 | 1, mid + 1, cr);

     tree[p] = tree[p << 1] + tree[p << 1 | 1];

}

void update2(int l, int r, int d, int p = 1, int cl = 1, int cr = n)

{

    if (cl >= l && cr <= r)

        return void((tree[p] += d \* (cr - cl + 1), mark[p] += d));

    // push\_down(p, cr - cl + 1);

    tree[p] += max(0, min(cr, r) - max(cl, l) + 1) \* d;

    int mid = (cl + cr) >> 1;

    if (mid >= l)

        update(l, r, d, p << 1, cl, mid);

    if (mid < r)

        update(l, r, d, p << 1 | 1, mid + 1, cr);

    // tree[p] = tree[p << 1] + tree[p << 1 | 1];

}

int main()

{

    ios::sync\_with\_stdio(false);

    cin >> n >> m;

    for (int i = 1; i <= n; ++i)

        cin >> a[i];

    build();

    while (m--)

    {

        int o, l, r, d;

        cin >> o >> l >> r;

        if (o == 1)

            cin >> d, update(l, r, d);

        else

            cout << query(l, r) << '\n';

    }

    return 0;

}