```cpp
//
//   There are N stones, numbered 1,2, ... N. For each i (1 ⩽ i ⩽ N), the height of Stone i is h[i].
//   Here h[1] < h[2] < [3] ... < h[N] holds.
//   There is a frog who is initially on Stone 1.
//   He will repeat the following action some number of times to reach Stone N:
//     - If the frog is currently on Stone i, jump to one of the following:
//         Stone i+1,i+2, ... N. Here, a cost of (h[i] - h[j])^2 + C is incurred, where j is the stone to land on.
//   Find the minimum possible total cost incurred before the frog reaches Stone N.
//
//     Time Complexity: O(N)
//

#include <bits/stdc++.h>
#define ll long long

using namespace std;

vector<ll> h, dp;
int n;
ll C;

struct Line {
        ll k, m, p;
        ll eval(ll x) const { return k * x + m; }
};

struct CHT {
        deque<Line> hull;
        static const ll inf = LLONG_MAX;

        ll div(ll a, ll b) { // floored division
                return a / b - ((a ^ b) < 0 && a % b); }

        bool intersect(Line &x, const Line &y) {
                if (x.k == y.k) x.p = x.m > y.m ? inf : -inf;
                else x.p = div(y.m - x.m, x.k - y.k);
                return x.p >= y.p;
        }

    // Add a line with a form of kx + m
        void add(ll k, ll m) {
                Line L = {k, m, 0};
                while ((int) hull.size() >= 2 && (intersect(L, hull.back()),
                        intersect(hull.back(), hull[(int) hull.size() - 2]), L.p < hull.back().p))
                        hull.pop_back();
                hull.push_back(L);
        }

        // query at point x equivalent to cht(x)
        ll query(ll x) {
                while ((int) hull.size() >= 2 && hull[0].eval(x) >= hull[1].eval(x))
                        hull.pop_front();
                return hull[0].eval(x);
        }
};

int main() {
    cin >> n >> C;

    h.resize(n);
    dp.resize(n);
    for (int i = 0; i < n; i++) cin >> h[i];

    CHT cht;
    auto insert = [&](int i) {
        cht.add(-2LL * h[i], (dp[i] + (h[i] * h[i])));
    };

    dp[0] = 0; insert(0);
    for (int i = 1; i < n; i++) {
        ll x = cht.query(h[i]);
        dp[i] = C + (h[i] * h[i]) + x;

        insert(i);
    }

    cout << dp[n-1] << endl;
    return 0;
}
```