### pollard pho

#include <bits/stdc++.h>   
#define sz(x) int((x).size())   
#define all(x) begin(x), end(x)   
   
using namespace std;   
template<class T>   
using vc = vector<T>;   
using ull = unsigned long long;   
using ll = long long;   
   
ull modmul(ull a, ull b, ull M) {   
 ll ret = a \* b - M \* ull(1.L / M \* a \* b);   
 return ret + M \* (ret < 0) - M \* (ret >= (ll)M);   
}   
   
ull modpow(ull b, ull e, ull mod) {   
 ull ans = 1;   
 for (; e; b = modmul(b, b, mod), e /= 2)   
 if (e & 1) ans = modmul(ans, b, mod);   
 return ans;   
}   
   
ull Qpow(ull b, int e) {   
 ull res = 1;   
 for (; e; b \*= b, e /= 2) if (e & 1) res \*= b;   
 return res;   
}   
   
bool isPrime(ull p) {   
 if (p == 2) return true;   
 if (p == 1 || p % 2 == 0) return false;   
 ull s = p - 1;   
 while (s % 2 == 0) s /= 2;   
 for (int i = 0; i < 15; ++i) {   
 ull a = rand() % (p - 1) + 1, tmp = s;   
 ull mod = modpow(a, tmp, p);   
 while (tmp != p - 1 && mod != 1 && mod != p - 1) {   
 mod = modmul(mod, mod, p);   
 tmp \*= 2;   
 }   
 if (mod != p - 1 && tmp % 2 == 0) return false;   
 }   
 return true;   
}   
   
ull pollard(ull n) {   
 auto f = [n](ull x) { return modmul(x, x, n) + 1; };   
 ull x = 0, y = 0, t = 30, prd = 2, i = 1, q;   
 while (t++ % 40 || \_\_gcd(prd, n) == 1) {   
 if (x == y) x = ++i, y = f(x);   
 if ((q = modmul(prd, max(x,y) - min(x,y), n))) prd = q;   
 x = f(x), y = f(f(y));   
 }   
 return \_\_gcd(prd, n);   
}   
   
vector<ull> factor(ull n) {   
 if (n == 1) return {};   
 if (isPrime(n)) return {n};   
 ull x = pollard(n);   
 auto l = factor(x), r = factor(n / x);   
 l.insert(l.end(), all(r));   
 return l;   
}   
   
int main() {   
#ifdef LOCAL   
 freopen("in.txt", "r", stdin);   
#endif   
 cin.tie(nullptr)->sync\_with\_stdio(false);   
 int n; cin >> n;   
 while (n--) {   
 ull x; cin >> x;   
 auto fac = factor(x);   
 map<ull, int> mp;   
 for (auto e: fac) {   
 ++mp[e];   
 }   
 ull ans = 1;   
 for (auto p: mp) {   
 ans \*= Qpow(p.first, p.second / 3);   
 }   
 cout << ans << '\n';   
 }   
}