```cpp
#include <bits/stdc++.h>

using namespace std;
using ll = long long;
using ld = long double;
using cx = complex<ld>;

ld pi = 3.14159265358979323846264338327950288841971;

vector<cx> dft(vector<cx> f, int n) {
  if (n == 1) return f;
  vector<cx> f0(n / 2), f1(n / 2);
  for (int i = 0; i < n / 2 && i * 2 < f.size(); i++) f0[i] = f[i * 2];
  for (int i = 0; i < n / 2 && i * 2 + 1 < f.size(); i++) f1[i] = f[i * 2 + 1];
  auto f0c = dft(f0, n / 2);
  auto f1c = dft(f1, n / 2);
  vector<cx> fc(n);
  for (int i = 0; i < n; i++) {
    cx zeta_n_i = cx(cos(2 * pi * i / n), sin(2 * pi * i / n));
    fc[i] = f0c[i % (n / 2)] + zeta_n_i * f1c[i % (n / 2)];
  }
  return fc;
}

vector<cx> idft(vector<cx> fc, int n) {
  auto fcc = dft(fc, n);
  vector<cx> swaped(n);
  for (int i = 0; i < n; i++) swaped[i] = fcc[(n - i) % n];
  vector<cx> ret(n);
  for (int i = 0; i < n; i++) ret[i] = swaped[i] / cx(n, 0);
  return ret;
}

vector<cx> mul(vector<cx> a, vector<cx> b) {
  int n = 1;
  while (n < a.size() + b.size() + 1) {
    n <<= 1;
  }
  vector<cx> cc(n);
  auto ac = dft(a, n);
  auto bc = dft(b, n);
  for (int i = 0; i < n; i++) cc[i] = ac[i] * bc[i];
  return idft(cc, n);
}

int main() {
  int n;
  cin >> n;
  vector<cx> a(n + 1), b(n + 1);
  for (int i = 1; i <= n; i++) cin >> a[i] >> b[i];
  auto c = mul(a, b);
  for (int i = 1; i <= 2 * n; i++) {
    cout << (int) round(c[i].real() + .1) << endl;
  }

  return 0;
}
```