```cpp
const double PI = acos(-1.0);
typedef complex<double> base;

void fft(vector<base> & a, bool invert) {
    int n = (int)a.size();
    for (int i = 1, j = 0; i<n; ++i) {
        int bit = n >> 1;
        for (; j >= bit; bit >>= 1)
            j -= bit;
        j += bit;
        if (i < j)
            swap(a[i], a[j]);
    }
    for (int len = 2; len <= n; len <<= 1) {
        double ang = 2 * PI / len * (invert ? -1 : 1);
        base wlen(cos(ang), sin(ang));
        for (int i = 0; i<n; i += len) {
            base w(1);
            for (int j = 0; j<len / 2; ++j) {
                base u = a[i + j], v = a[i + j + len / 2] * w;
                a[i + j] = u + v;
                a[i + j + len / 2] = u - v;
                w *= wlen;
            }
        }
    }
    if (invert)
    for (int i = 0; i<n; ++i)
        a[i] /= n;
}

vi mult(vi a, vi b) {
    vector<base> fa(a.begin(), a.end()), fb(b.begin(), b.end());
    size_t n = 1;
    while (n < max(a.size(), b.size()))  n <<= 1;
    n <<= 1;
    fa.resize(n), fb.resize(n);
    fft(fa, false), fft(fb, false);
    for (size_t i = 0; i<n; ++i)
        fa[i] *= fb[i];
    fft(fa, true);
    vi res;
    res.resize(n);
    for (size_t i = 0; i<n; ++i)
        res[i] = int(fa[i].real() + 0.5);
    return res;
}
```