```cpp
namespace Fourier_transform {

  vector<cdouble> omegas, iomegas;

  inline int bit_reverse(int x, int digit) {
    int ret = digit ? x & 1 : 0;
    Loop(i, digit - 1) { ret <<= 1; x >>= 1; ret |= x & 1; }
    return ret;
  }

  inline void make_omegas(int n) {
    if (omegas.size() != n) {
      omegas.resize(n);
      Loop(i, n) omegas[i] = exp(cdouble({ 0, 2 * PI * i / n }));
    }
  }

  inline void make_iomegas(int n) {
    if (iomegas.size() != n) {
      iomegas.resize(n);
      Loop(i, n) iomegas[i] = exp(cdouble({ 0, -2 * PI * i / n }));
    }
  }

  // a.size() should be 2^digit
  vector<cdouble> FFT(const vector<cdouble> a) {
    int n = int(a.size());
    int digit = int(rndf(log2(n)));
    vector<cdouble> ret = a;
    make_omegas(n);
    Loop(i, n) {
      int j = bit_reverse(i, digit);
      if (j > i) swap(ret[i], ret[j]);
    }
    Loop(i, digit) {
      int j = 0, m = 1 << i, mw = (digit - i - 1);
      Loop(group_id, n >> (i + 1)) {
        Loop(k, m) {
          cdouble x = ret[j] + omegas[k << mw] * ret[j + m];
          cdouble y = ret[j] - omegas[k << mw] * ret[j + m];
          ret[j] = x; ret[j + m] = y;
          ++j;
        }
        j += m;
      }
    }
    return ret;
  }

  // f.size() should be 2^digit
  vector<cdouble> IFFT(const vector<cdouble>& f) {
    int n = int(f.size());
    int digit = int(rndf(log2(n)));
    vector<cdouble> ret = f;
    make_iomegas(n);
    Loopr(i, digit) {
      int j = 0, m = 1 << i, mw = (digit - i - 1);
      Loop(group_id, n >> (i + 1)) {
        Loop(k, m) {
          cdouble q = (ret[j] + ret[j + m]) * 0.5;
          cdouble r = (ret[j] - ret[j + m]) * 0.5 * iomegas[k << mw];
          ret[j] = q; ret[j + m] = r;
          ++j;
        }
        j += m;
      }
    }
    Loop(i, n) {
      int j = bit_reverse(i, digit);
      if (j > i) swap(ret[i], ret[j]);
    }
```

```
72      return ret;
73    }
74
75    // a.size() = b.size() should be 2^digit
76    vector<cdouble> mul_convolution(const vector<cdouble> &a, const vector<cdouble> &b) {
77      int n = int(a.size());
78      vector<cdouble> ret;
79      vector<cdouble> g = FFT(a), h = FFT(b);
80      Loop(i, n) g[i] *= h[i];
81      ret = IFFT(g);
82      return ret;
83    }
84
85    int legal_size_of(int n) {
86      int ret = 1 << (int)log2(n);
87      if (ret < n) ret <<= 1;
88      return ret;
89    }
90  }
91
92  using namespace Fourier_transform;
```