```
namespace Fourier_transform {
3
      vector < cdouble > omegas, iomegas;
4
5
      inline int bit_reverse(int x, int digit) {
        int ret = digit ? \times \& 1 : 0;
7
        Loop(i, digit - 1) { ret \leq 1; x > 1; ret |= x \& 1; }
8
        return ret;
9
      }
10
11
      inline void make_omegas(int n) {
12
        if (omegas.size() != n) {
13
          omegas.resize(n);
14
          Loop(i, n) omegas[i] = \exp(\operatorname{cdouble}(\{0, 2 * PI * i / n \}));
15
16
      }
17
18
      inline void make_iomegas(int n) {
19
        if (iomegas.size() != n) {
20
          iomegas.resize(n);
21
          Loop(i, n) iomegas[i] = exp(cdouble({0, -2 * PI * i / n}));
22
        }
     }
23
24
25
      // a.size() should be 2<sup>digit</sup>
26
      vector < cdouble > FFT (const vector < cdouble > a) {
27
        int n = int(a.size());
28
        int digit = int(rndf(log2(n)));
29
        vector<cdouble> ret = a;
30
        make_omegas(n);
31
        Loop(i, n) {
32
          int j = bit_reverse(i, digit);
33
          if (j > i) swap(ret[i], ret[j]);
34
35
        Loop(i, digit) {
          int j = 0, m = 1 << i, mw = (digit - i - 1);
36
37
          Loop (group_id, n \gg (i + 1)) {
38
            Loop(k, m) {
39
              cdouble x = ret[j] + omegas[k << mw] * ret[j + m];
40
              cdouble y = ret[j] - omegas[k << mw] * ret[j + m];
41
              ret[j] = x; ret[j + m] = y;
42
              ++j;
43
44
            j += m;
          }
45
46
47
        return ret;
48
49
50
      // f.size() should be 2<sup>digit</sup>
51
      vector<cdouble> IFFT(const vector<cdouble>& f) {
52
        int n = int(f.size());
53
        int digit = int(rndf(log2(n)));
54
        vector<cdouble> ret = f;
55
        make_iomegas(n);
56
        Loopr(i, digit) {
          int j = 0, m = 1 << i, mw = (digit - i - 1);
57
58
          Loop(group_id, n \gg (i + 1)) {
59
            Loop(k, m)
60
              cdouble q = (ret[j] + ret[j + m]) * 0.5;
61
              cdouble r = (ret[j] - ret[j + m]) * 0.5 * iomegas[k \leftleft mw];
62
              ret[j] = q; ret[j + m] = r;
63
              ++ j;
64
65
            j += m;
66
          }
67
68
        Loop(i, n) {
69
          int j = bit_reverse(i, digit);
70
          if (j > i) swap(ret[i], ret[j]);
71
```

```
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];
        ret = IFFT(g);
81
82
        return ret;
      }
83
84
85
      int legal_size_of(int n) {
86
        int ret = 1 \ll (int) \log_2(n);
        if (ret < n) ret <<= 1;</pre>
87
88
        return ret;
89
      }
   }
90
91
92 using namespace Fourier_transform;
```