

```
1 namespace Zeta_and_Mobius_transform {
2
3 // f.size() should be 2^digit, ret will assemble value from subsets
4 vll Zeta_trans(vll f) {
5     int n = f.size();
6     int digit = int(rndf(log2(n)));
7     vll ret = f;
8     Loop(i, digit) {
9         int x = 1 << i;
10        Loop(j, n) {
11            if (j & x) ret[j] += ret[j ^ x];
12        }
13    }
14    return ret;
15 }
16
17 // g.size() should be 2^digit, ret will disassemble value to subsets
18 vll Mobius_trans(vll g) {
19     int n = g.size();
20     int digit = int(rndf(log2(n)));
21     vll ret = g;
22     Loop(i, digit) {
23         int x = 1 << i;
24         Loop(j, n) {
25             if (j & x) ret[j] -= ret[j ^ x];
26         }
27     }
28     return ret;
29 }
30
31 // f.size() should be 2^digit, ret will assemble value from supersets
32 vll Zeta_trans_rev(vll f) {
33     int n = f.size();
34     int digit = int(rndf(log2(n)));
35     vll ret = f;
36     Loop(i, digit) {
37         int x = 1 << i;
38         Loop(j, n) {
39             if (!(j & x)) ret[j] += ret[j | x];
40         }
41     }
42     return ret;
43 }
44
45 // g.size() should be 2^digit, ret will disassemble value to supersets
46 vll Mobius_trans_rev(vll g) {
47     int n = g.size();
48     int digit = int(rndf(log2(n)));
49     vll ret = g;
50     Loop(i, digit) {
51         int x = 1 << i;
52         Loop(j, n) {
53             if (!(j & x)) ret[j] -= ret[j | x];
54         }
55     }
56     return ret;
57 }
58
59 int legal_size_of(int n) {
60     int ret = 1 << (int)log2(n);
61     if (ret < n) ret <= 1;
62     return ret;
63 }
64 }
65
66 using namespace Zeta_and_Mobius_transform;
```