```cpp
namespace Zeta_and_Mobius_transform {

  // f.size() should be 2^digit, ret will assemble value from subsets
  vll Zeta_trans(vll f) {
    int n = f.size();
    int digit = int(rndf(log2(n)));
    vll ret = f;
    Loop(i, digit) {
      int x = 1 << i;
      Loop(j, n) {
        if (j & x) ret[j] += ret[j ^ x];
      }
    }
    return ret;
  }

  // g.size() should be 2^digit, ret will disassemble value to subsets
  vll Mobius_trans(vll g) {
    int n = g.size();
    int digit = int(rndf(log2(n)));
    vll ret = g;
    Loop(i, digit) {
      int x = 1 << i;
      Loop(j, n) {
        if (j & x) ret[j] -= ret[j ^ x];
      }
    }
    return ret;
  }

  // f.size() should be 2^digit, ret will assemble value from supersets
  vll Zeta_trans_rev(vll f) {
    int n = f.size();
    int digit = int(rndf(log2(n)));
    vll ret = f;
    Loop(i, digit) {
      int x = 1 << i;
      Loop(j, n) {
        if (!(j & x)) ret[j] += ret[j | x];
      }
    }
    return ret;
  }

  // g.size() should be 2^digit, ret will disassemble value to supersets
  vll Mobius_trans_rev(vll g) {
    int n = g.size();
    int digit = int(rndf(log2(n)));
    vll ret = g;
    Loop(i, digit) {
      int x = 1 << i;
      Loop(j, n) {
        if (!(j & x)) ret[j] -= ret[j | x];
      }
    }
    return ret;
  }

  int legal_size_of(int n) {
    int ret = 1 << (int)log2(n);
    if (ret < n) ret <<= 1;
    return ret;
  }
}

using namespace Zeta_and_Mobius_transform;
```