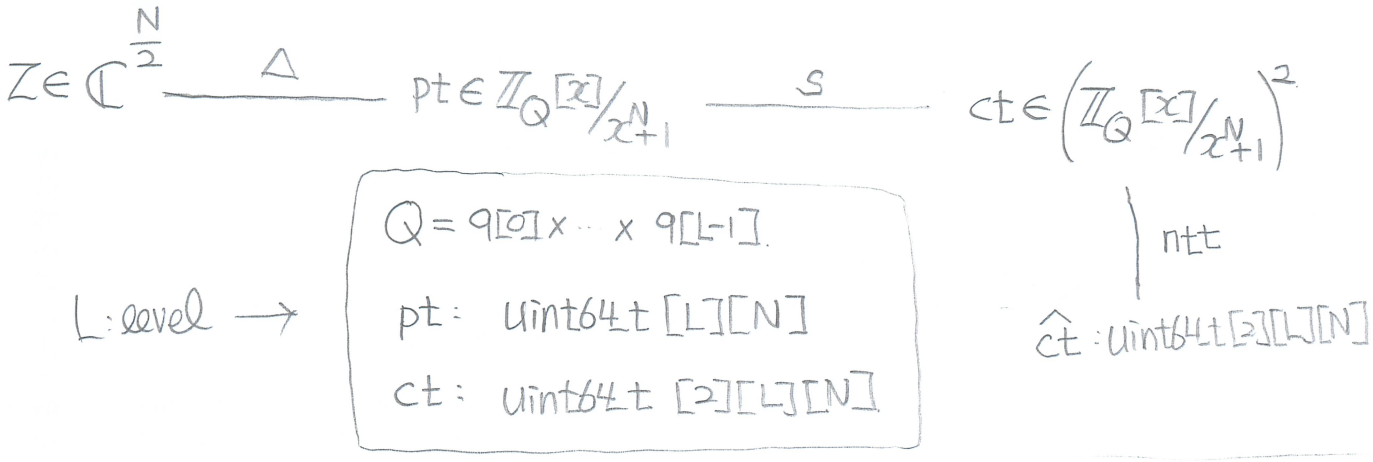


```
template < int N, int logN, int L>
void encode( const double zr[N/2],
             const double zi[N/2], uint64_t Delta,
             const uint64_t q[L], uint64_t pt[L][N]){
    double m[N]; ifft<N, logN>( zr, zi, m);
    for(int i=0; i<N; i++){
        uint64_t pti= uint64_t((m[i]>0?m[i]:-m[i])*Delta)+0.5);
        for(int j=0; j<L; j++){
            pt[j][i] = pti % q[j];
            if(m[i] <0) pt[j][i] = q[j] - pt[j][i];
    }   }
}
```

```
template < int N, int logN, int L>
void decode ( const uint64_t pt[L][N], uint64_t Delta,
    const uint64_t q[L], double zr[N/2], double zi[N/2] ){
```

Q/2

```
BigInt Qhalf; { BigInt Q(1); Q[0]=1; for(int i=0; i<L; i++){
                BigInt temp(Q); temp.mul(q[i], Q); }
              uint64_t r; Q.div(2, Qhalf, r); }
```

```
double m[N];
for(int j=0; j<N; j++){ uint64_t a[L]; for(int i=0; i<L; i++) a[i]= pt[i][j];
    BigInt A; icrt(q,a,A);
    if(A>=Qhalf){ BigInt temp(A); Q.sub(temp, A);
                  m[j] = - A.to_real(); }
    else          m[j] =   A.to_real();
}
```

```
    fft<N, logN>(m, zr, zi);
}
```

```
template<int N>
void keygen(int h, int s[N]){
    for(int i=0; i<N; i++) s[i]=0;
    for(int i=0; i<h; i++){
        int j= rand()%N;
        while(s[j]!=0)
            j= rand()%N;
        s[j]= (rand()%2==0)? 1: -1;
    }
}
```

```
template<int N, int L>
void enc(const uint64_t pt[L][N], const int s[N],
         const uint64_t q[L], uint64_t ctbar[2][L][N]){
  int e[N]; for(int i=0; i<N; i++) e[i]= (rand()%17)-8;
  for(int j=0; j<L; j++){
    uint64_t shat[N];
    for(int i=0; i<N; i++){
      shat[i]= (q[j]+s[i])%q[j];
      ctbar[i][j][i]= rand_uint64()%q[j];
      ctbar[0][j][i]= (q[j]+e[i]+pt[j][i])%q[j];
    }
    ntt<N>(ctbar[0][j], q[j]);
    ntt<N>(ctbar[1][j], q[j]);
    ntt<N>(shat, q[j]);
    for(int i=0; i<N; i++)
      ctbar[0][j][i]= (ctbar[0][j][i] + q[j]
                       - mul_mod(shat[i], ctbar[1][j][i], q[j]))
                      % q[j];
  }
}
```

$\sigma^2 = \frac{8}{17} \cdot 3 \cdot \frac{1}{17} = 24$

ct[0] = pt + e
ct[1] = rand

ct[0] = pt + e - s*ct[1]

```
template<int N, int L>
void dec(const uint64_t cthat[2][L][N], const uint64_t q[L],
         const int s[N], uint64_t pt[L][N]) {
  for(int j=0; j<L; j++) {
    uint64_t shat[N];
    for(int i=0; i<N; i++)
      shat[i] = (q[j] + s[i]) % q[j];
    ntt<N>(shat, q[j]);
    for(int i=0; i<N; i++)
      pt[j][i] = (cthat[0][j][i] +
                  mul_mod(cthat[1][j][i], shat[i], q[j])
                  % q[j];
    intt<N>(pt[j], q[j]);
  }
}
```

$\hat{pt} = \hat{ct}_0 + \hat{s} \cdot \hat{ct}_1$

RS: $a \in \mathbb{Z}_{q_0} \times \cdots \times \mathbb{Z}_{q_{L-1}} \longrightarrow \tilde{a} = \left\lfloor \frac{a}{q_{L-1}} \right\rceil \in \mathbb{Z}_{q_0} \times \cdots \times \mathbb{Z}_{q_{L-2}}$.

$a = q_{L-1} A + a[L-1]$

$\left\lfloor \frac{a}{q_{L-1}} \right\rceil = A + \left\lfloor \frac{a[L-1]}{q_{L-1}} \right\rceil$

$\tilde{a}[j] \equiv A + \left\lfloor \frac{a[L-1]}{q_{L-1}} \right\rceil$

$\equiv (a[j] - a[L-1]) \cdot \text{inv\_mod}(q[L-1], q[j]) + \left\lfloor \frac{a[L-1]}{q_{L-1}} \right\rceil \pmod{q[j]}$

```
template<int N, int L>
void RS(const uint64_t q[L], const uint64_t pt[L][N], uint64_t pt_[L-1][N]) {
  uint64_t qinv[L-1];
  for(int j=0; j<L-1; j++) qinv[j] = inv_mod(q[L-1], q[j]);

  for(int i=0; i<N; i++) {
    uint64_t r = (pt[L-1][i] > (q[L-1]/2)) ? 1 : 0;
    for(int j=0; j<L-1; j++)
      pt_[j][i] = (mul_mod((pt[j][i] + q[j] - (pt[L-1][i] % q[j])) % q[j],
                           qinv[j], q[j]) + r) % q[j];
  }
}
```

public key generation

```
template<int N, int L>
void keygen( const int s[N], const uint64_t q[L],
             uint64_t pk[2][L][N] ) {
    uint64_t zero[L][N];
    for(int i=0; i<L; i++)
    for(int j=0; j<N; j++) zero[i][j]= 0;
    enc<N,L>(zero, s, q, pk);
}
```

pk = enc(0, s)

encryption with public key

```
template<int N, int L>
void enc( const uint64_t pt[L][N],
          const uint64_t pk[2][L][N], const uint64_t q[L],
          uint64_t ct[2][L][N] ) {
    int e0[N], e1[N], v[N];
    for(int i=0; i<N; i++){
        e0[i]= (rand()%17) - 8;
        e1[i] = (rand()%17) - 8;   int r=rand()%4;
        v[i] = (r==0)?(-1):((r==1)? 1 : 0);
    }
    for(int j=0; j<L; j++){
        uint64_t v̂[N], pt̂[N], ê0[N], ê1[N];
        for(int i=0; i<N; i++){
            v̂[i] = (q[j]+v[i]) % q[j];
            pt̂[i] = (q[j] + pt[j][i]) % q[j];
            ê0[i] = (  "  + e0[i] ) % q[j];
            ê1[i] = (  "  + e1[i] ) % q[j];
        }
        ntt<N>(v̂, q[j]);   — pt̂ — ê0 — ê1
        for(int i=0; i<N; i++){
            ct[0][j][i] = (mul_mod(v̂[i], pk[0][j][i], q[j]) + pt̂[i] + ê0[i]) % q[j];
            ct[1][j][i] = (mul_mod(v̂[i], pk[1][j][i], q[j])        + ê1[i]) % q[j];
        }
    }
}
```

ct = v×pk + (pt+e0, e1)

e0, e1 : U[-8,8] (discrete)

v : ZO(-1, 1)

v̂, pt̂, ê0, ê1 계산

ĉt = v̂ ⊙ pk + [pt̂ + ê0, ê1]