NumberTheory.h

```
#include <inttypes.h>

extern "C" {
    void mul_asm(uint64_t a, uint64_t b,
                 uint64_t* hi, uint64_t* lo);

    uint64_t mod_asm(uint64_t hi, uint64_t lo, uint64_t a);
}
```

```
uint64_t mul_mod(uint64_t a, uint64_t b, uint64_t q){   <- a·b (mod q)
    uint64_t hi, lo; mul_asm(a,b,&hi,&lo);
    return mod_asm(hi,lo,q);
}
```

```
uint64_t power_mod(uint64_t a, uint64_t n, uint64_t q){   <- a^n (mod q)
    uint64_t m=a, out=1;
    while(n>0){
        if(n%2==1) out=mul_mod(out,m,q);
        m = mul_mod(m,m,q);
        n >>=1;
    }
    return out;
}
```

```
uint64_t inv_mod(uint64_t a, uint64_t q){   <- a^{-1} ≡ a^{q-2} (mod q)
    return power_mod(a, q-2, q);
}
```

```
uint64_t rand_uint64();
```

```
bool isprime(uint64_t n){
    if(n<=1) return false;
    if(n<=3) return true;
    if(n%2==0 || n%3==0) return false;
    for(uint64_t i=5; i*i<=n; i=i+6)
        if(n%i==0 || n%(i+2)==0)
            return false;

    return true;
}
```

$n \equiv 0,1,2,3,4,5 \pmod 6$

$n = i \cdot (n/i)$

$5 < i < \sqrt{n}$ 이고 $i \equiv 5,1 \pmod 6$

```
void findPrimeFactors(uint64_t n, std::unordered_set<uint64_t>& S){
    while(n%2==0){
        S.insert(2);
        n=n/2;
    }
    while(n%3==0){
        S.insert(3);
        n=n/3;
    }
    for(uint64_t i=5; i*i<=n; i=i+6){
        while(n%i==0){
            S.insert(i);
            n=n/i;
        }
        while(n%(i+2)==0){
            S.insert(i+2);
            n=n/(i+2);
        }
    }
    if(n>1)
        S.insert(n);
}
```

이후 n은 2의 배수, 3의 배수가 아니므로.

n을 나누는 소인수는 $\equiv 1,5 \pmod 6$

```
uint64_t findPrimitive(uint64_t n){
    static std::map<uint64_t, uint64_t> table;
    static bool table_initialized = false;
    if(table_initialized == false){
        uint64_t q[60] = { ========== };

        uint64_t psi[60] = { ========== };
        for(int i=0; i<60; i++)
            table[q[i]] = psi[i];
    }
    if(table.find(n)!=table.end()) return table[n];
    else{
```

find_RNS_primes()에서 미리 계산해 둔 소수들

zn_primroot()로 미리 계산

table에 있음

```
if( isPrime(n)==false)
    return -1;
unordered_set<uint64_t> S;
findPrimeFactors(n-1, S);
for(uint64_t r=2; r<=n-1; r++){
    bool flag= false;
    for(auto it=S.begin(); it!=S.end(); it++){
        if( power_mod(r, (n-1)/(*it), n)==1){
            flag= true;
            break;
        }
    }
    if(flag== false){ table[n]=r;
        return r; }
}
}
```

$n-1 = P_1^{R_1} P_2^{R_2} P_3^{R_3}$

r: primitive root $\Leftrightarrow$ $r^{\frac{n-1}{P_1}}, r^{\frac{n-1}{P_2}}, r^{\frac{n-1}{P_3}} \neq 1$

없으면 새로 계산하고 table에 추가

```
template<int N>
void ntt(uint64_t a[N], uint64_t p)
    uint64_t psi = findPrimitive(p)%p;
    psi = power_mod(psi, (p-1)/(2*N), p);
    uint64_t Psi[N], PsiRev[N]; Psi[0]=1;
    for(int i=1; i<N; i++)
        Psi[i] = mul_mod(psi, Psi[i-1], p);
    bitReverse<N>(Psi, PsiRev);
    for(int m=1, t=N/2; m<N; m*=2, t/=2){
        for(int i=0; i<m; i++){
            uint64_t* ae = a + 2*i*t;
            uint64_t* ao = ae + t;
            for(int k=0; k<t; k++){
                uint64_t U = mod_mul(ao[k], PsiRev[m+i], p);
                ao[k] = (ae[k]+p - U)%p;
                ae[k] = (ae[k] + U)%p;
            }
        }
    }
```

$$\hat{a}[bitR(j)] = \sum_{i=0}^{N-1} a[i]\,\zeta^{(2j+1)i}, \quad j=0,\cdots,N-1$$

```
template<int N>
void intt(uint64_t a[N], uint64_t p)
    uint64_t psi = power_mod(findPrimitive(p)%p, (p-1)/(2*N), p);
    psi = inv_mod(psi, p);
    for(int m=N/2, t=1; m>0; m/=2, t*=2){
        for(int i=0; i<m; i++){
            uint64_t* ae = a + 2*i*t; uint64_t* ao = ae + t;
            for(int k=0; k<t; k++){
                uint64_t U = (ae[k] + p - ao[k])%p;
                ae[k] = (ae[k] + ao[k])%p;
                ao[k] = mod_mul(U, PsiRev[m+i], p);
            }
        }
    }
    uint64_t Ninv = inv_mod(N, p);
    for(int i=0; i<N; i++) a[i] = mod_mul(a[i], Ninv, p);
}
```

$$a[i] = \frac{1}{N}\sum_{j=0}^{N-1} \hat{a}[bitR(j)]\,\zeta^{-(2j+1)i}$$

```
template<int L>
void icrt( const uint64_t q[L],
           const uint64_t a[L], BigInt& A) {
```

Q

```
BigInt Q(L); Q[0]=1;
for(int i=0; i<L; i++){
    BigInt temp; Q.mul(q[i], temp);
    Q=temp;}
Q.print("Q");
```

$$A = a[0]\,\frac{Q}{q[0]}\cdot inv\left(\frac{Q}{q[0]}, q[0]\right) + \cdots + a[L-1]\cdot\frac{Q}{q[L-1]}\cdot inv\left(\frac{Q}{q[L-1]}, q[L-1]\right) \in [0,\ Q\cdot L)$$

```
A.resize(L+1);
for(int i=0; i<L; i++){
    BigInt Qoverq; uint64_t r;
    Q.div(q[i], Qoverq, r);
    BigInt temp1; Qoverq.mul( mul_mod(a[i],
                  inv_mod(Qoverq.mod(q[i],q[i]),q[i]),q[i]),
                  temp1);
    BigInt temp2=A;  temp1.add(temp2, A);
}
```

A∈[0,Q)

```
BigInt Q2; Q.mul(2, Q2);
BigInt Q4; Q2.mul(2, Q4);
BigInt Q8; Q4.mul(2, Q8);
BigInt Q16; Q8.mul(2, Q16);
while( A>=Q16){ BigInt temp; A.sub(Q16, temp); A=temp;}
```

Q8 Q4 Q2 Q

Q8 Q4 Q2 Q

```
}
```

```
template<int N, int L>
void rot(const uint64_t q[L], const uint64_t a[L][N],
         uint64_t a_rot[L][N]){

  for(int i=0; i<L; i++)
  for(int j=0; j<N; j++){
    int Aj = (5*j) / N;
    int rj = (5*j) % N;
    a_rot[i][rj]
      = (Aj%2==0)? a[i][j]
                 :((q[i] - a[i][j])%q[i]);
  }
}
```

$$a(x) = \sum_{j=0}^{N-1} a[i][j]\, x^j$$

$$a(x^5) = \sum_{j=0}^{N-1} a[i][j]\, x^{5j}$$

$$= \sum_{j=0}^{N-1} a[i][j]\, x^{A_j N + r_j}$$

$$= \sum_{j=0}^{N-1} a[i][j]\,(-1)^{A_j}\, x^{r_j}$$

```
template<int N, int L>
void rot_hat(const uint64_t â[L][N],
             uint64_t â_rot[L][N]){

  int Normal[N]; for(int i=0; i<N; i++) Normal[i] = i;
  int Reverse[N]; bitReverse<N>(Normal, Reverse);

  for(int j=0; j<N; j++){
    int bRj = Reverse[j];
    int rj = (10*bRj+5) % (2*N);
    for(int i=0; i<L; i++)
      â_rot[i][j]
        = â[i][Reverse[(rj-1)/2]];
  }
}
```

$$\hat{a}[j] = a(\zeta^{2\cdot bitR(j)+1})$$

$$\hat{a}_{rot}[j] = a_{rot}(\zeta^{2\cdot bR(j)+1})$$

$$= a(\zeta^{10\cdot bR(j)+5})$$

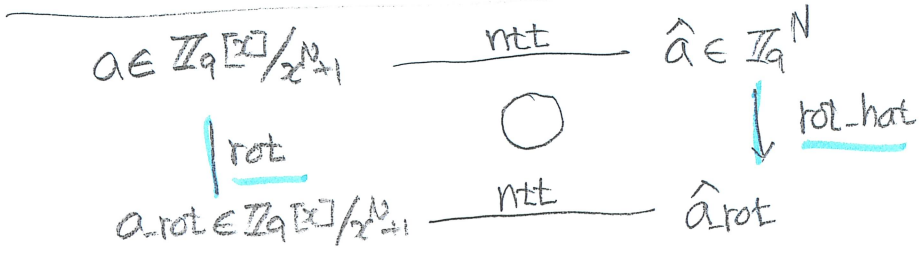
$$= a(\zeta^{2N A_j + r_j})$$

$$= a(\zeta^{r_j})$$

$$= a(\zeta^{2\cdot(\frac{r_j-1}{2})+1})$$

$$= a(\zeta^{2\cdot bR(bR(\frac{r_j-1}{2}))+1})$$

$$= \hat{a}[bR(\tfrac{r_j-1}{2})]$$



```
template<int L>
void find_RNS_primes(uint64_t Delta, int N, uint64_t q[L]){
```

전제: $\Delta = 2^{50}$ or 2^{60}, $N \le 2^{17}$, $L < 30$

prime 들을 미리계산해 둠.

```
Delta: 2^50; N: 2^17;
q: next_prime(Delta); qL: [];
for i:1 thru 30 do
(   while mod(q,2*N) > 1 do (
      q: next_prime(q + 2*N - mod(q, 2*N))),
    qL: append(qL, [q]);
    q: next_prime(q + 2*N - mod(q, 2*N))
);
makelist(primep(qL[i]), i, 1, 30);
makelist(mod(qL[i], 2*N), i, 1, 30);
float(qL/Delta);
```

$\frac{q}{\Delta} \le 1 + 10^{-7}$ ($\Delta = 2^{50}$)

$1 + 10^{-10}$ ($\Delta = 2^{60}$)

```
if( Delta==(1ULL<<50) && N<=(1<<17) && L<=30){
    uint64_t q_save[30]={1125899915231233ULL,... };
    for(int i=0; i<L; i++)
        q[i]=q_save[i];
}
else if( Delta==(1ULL<<60) && N<=(1<<17) && L<=30){
    ==========
}
else{
    int count=0;
    uint64_t a=Delta;
    if((a-1)%(2*N)==0 && isprime(a)){
        q[count]=a;
        count++;
    }
}
}
```