polyn(FFT,NTT)

```cpp
#include <bits/stdc++.h>
#define int long long
#define endl '\n'
#define LL __int128
using namespace std;
int qpow(int a, int b, int p) {int ret = 1; for(a %= p; b; b >>= 1, a = a * a % p) if(b & 1) ret = ret * a % p; return ret; }
int qpow(int a,int b) {int ret = 1; for(; b; b >>= 1, a *= a) if(b & 1) ret *= a; return ret; }
int gcd(int x,int y) {return y ? gcd(y, x % y) : x; }
pair<int,int> exgcd(int a,int b) { if(!b) return {1, 0}; pair<int,int> ret = exgcd(b, a % b); return {ret.second, ret.first - a / b * ret.second }; }
int lcm(int x,int y){ return x / gcd(x, y) * y; }
namespace polyn {
    const int N = 1 << 21;
    const int P = 998244353;
    const int G = 3;
    const int GI = 332748118;
    int inv;
    int rev[N];
    struct Z {
        double a, b;
        Z(const double &_a = 0, const double &_b = 0) {
            a = _a, b = _b;
        }
        Z operator + (const Z &x) {
            return Z(a + x.a, b + x.b);
        }
        Z operator - (const Z &x) {
            return Z(a - x.a, b - x.b);
        }
        Z operator * (const Z &x) {
            return Z(a * x.a - b * x.b, a * x.b + b * x.a);
        }
    };
    const double PI = acos(-1);
    void init(const int &n, const int &pw) {
        for (int i = 0; i < n; i++) {
            rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (pw - 1));
        }
        inv = qpow(n, P - 2, P);
    }
    void FFT(Z *z, int n, bool T) {
        for (int i = 0; i < n; i++) {
            if (rev[i] > i) {
                swap(z[rev[i]], z[i]);
            }
        }
        for (int len = 2; len <= n; len <<= 1) {
            Z Wn = Z(cos(2.0 * PI / n), (T ? 1 : -1) * sin(2.0 * PI / n));
            for (int l = 0, r = len - 1; r < n; l += len, r += len) {
                Z w = Z(1, 0);
                for (int k = l, mid = l + (len >> 1); k < mid; k++, w = w * Wn) {
                    Z x = z[k], y = w * z[k + (len >> 1)];
                    z[k] = x + y, z[k + (len >> 1)] = x - y;
                }
            }
        }
        if (T) return;
        for (int i = 0; i < n; i++) {
            z[i].a /= n, z[i].b /= n;
        }
    }
    void NTT(int *z, int n, bool T) {
        for (int i = 0; i < n; i++) {
            if (rev[i] > i) {
                swap(z[rev[i]], z[i]);
            }
        }
        for (int len = 2; len <= n; len <<= 1) {
            int Wn = qpow(T ? G : GI, (P - 1) / len, P);
            for (int l = 0, r = len - 1; r < n; l += len, r += len) {
                int w = 1;
                for (int k = l, mid = l + (len >> 1); k < mid; k++, w = w * Wn % P) {
                    int x = z[k], y = w * z[k + (len >> 1)] % P;
                    z[k] = (x + y) % P, z[k + (len >> 1)] = (x - y + P) % P;
                }
            }
        }
        if (T) return;
        for (int i = 0; i < n; i++) {
            z[i] = z[i] * inv % P;
        }
    }
}
const int mod = 998244353;
const int _N = 1 << 21;
int _a[_N], _b[_N];
void cdq(int *f, int *g, int l, int r) {
    if (l == r) {
        return;
    }
    int mid = l + r >> 1;
    cdq(f, g, l, mid);
    for (int i = l; i <= mid; i++) {
        _a[i - l] = f[i];
    }
    for (int i = mid + 1; i <= r; i++) {
        _a[i - l] = 0;
    }
    for (int i = l; i <= r; i++) {
        _b[i - l] = g[i - l];
    }
    int n = 1, pw = 0;
    while (n < r - l + 1) {
        n <<= 1, pw++;
    }
    polyn :: init(n, pw);
    polyn :: NTT(_a, n, 1);
    polyn :: NTT(_b, n, 1);
    for (int i = 0; i < n; i++) {
        _a[i] = _a[i] * _b[i] % polyn :: P;
    }
    polyn :: NTT(_a, n, 0);
    for (int i = mid + 1; i <= r; i++) {
        f[i] += _a[i - 1];
        f[i] %= mod;
    }
    cdq(f, g, mid + 1, r);
}
const int N = 5 + 1e6;
int g[N], f[N];
signed main() {
    ios :: sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int n; cin >> n;
    for (int i = 1; i < n; i++) {
        cin >> g[i];
    }
    f[0] = 1;
    int _n = 1;
    while (_n < n) {
        _n <<= 1;
    }
    cdq(f, g, 0, _n);
    for (int i = 0; i < n; i++) {
        cout << f[i] << " ";
    }
    cout << endl;
}
```