FFT

```
#include <bits/stdc++.h>
using namespace std;

typedef pair<int, int> PII;
typedef long long ll;
const int INF = 0x3f3f3f3f;

const int N = 1e6 + 10;
const double PI = acos(-1);
int n, m;
struct Complex{
    double x, y;
    Complex operator+ (const Complex t) const{
        return {x + t.x, y + t.y};
    }
    Complex operator- (const Complex t) const{
        return {x - t.x, y - t.y};
    }
    Complex operator* (const Complex t) const{
        return {x * t.x - y * t.y, x * t.y + y * t.x};
    }
}a[N], b[N];
int rev[N], bit, tot;

void FFT(Complex a[], int inv){
    for(int i = 0; i < tot; i++){
        if(i < rev[i]) swap(a[i], a[rev[i]]);
    }
    for(int mid = 1; mid < tot; mid <<= 1){
        auto w1 = Complex({cos(PI / mid), inv * sin(PI / mid)});
        for(int i = 0; i < tot; i += mid * 2){
            auto wk = Complex({1, 0});
            for(int j = 0; j < mid; j++, wk = wk * w1){
                auto x = a[i + j], y = wk * a[i + j + mid];
                a[i + j] = x + y, a[i + j + mid] = x - y;
            }
        }
    }
}

int main(){
    scanf("%d%d", &n, &m);
    for(int i = 0; i <= n; i++) scanf("%lf", &a[i].x);
    for(int i = 0; i <= m; i++) scanf("%lf", &b[i].x);
    while((1 << bit) < n + m + 1) bit++;
    tot = 1 << bit;
    for(int i = 0; i < tot; i++){
        rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (bit - 1));
    }
    FFT(a, 1), FFT(b, 1);
```

```
    for(int i = 0; i < tot; i++) a[i] = a[i] * b[i];
    FFT(a, -1);
    for(int i = 0; i <= n + m; i++){
        printf("%d ", (int)(a[i].x / tot + 0.5));
    }
    return 0;
}
```