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

namespace FFT {
  typedef complex<double> Base;

  void rec_fft( vector<Base> &a ,bool invert) {
    int n = a.size() ;
    if( n == 1 ) return ;
    for( auto w : a ) cout << w << " " ; cout << endl;
    vector<Base> a0( n/2 , 0),a1 ( n/2 , 0);
    for(int i = 0;i<n/2;i++)
      a0[i] = a[i<<1],a1[i] = a[i<<1|1];
    double ang = 2 * M_PI /n * ( invert ? -1 : 1 );
    rec_fft ( a0 , invert), rec_fft ( a1 ,invert) ;
    Base w ( 1 ),wn ( cos ( ang ) , sin ( ang ) ) ;
    for( int i = 0; i < n/2; i ++ ) {
      a[i] = a0[i] + w * a1[i];
      a[i + n/2] = a0[i] - w * a1[i];
      if ( invert )
        a[i] /= 2,a[i + n/2] /= 2;
      w *= wn;
    }
  }

  void fft ( vector< Base> &a , bool invert ) {
    int n= a.size() ;
    double
  }
};

int main() {
  int n,m;cin >> n >> m;
  vector<int> v0 ( n , 0 ),v1 ( n , 0 );
  for ( int i = 0;i<n;i++) cin >> v0[i];
  for ( int i = 0;i<m;i++) cin >> v1[i];
  reverse(v1.begin(),v1.end());
  reverse(v0.begin(),v0.end());
  int sz = 1,x = 0;
  while ( sz < max( n,m ) ) sz <<= 1, x ++ ;
  sz <<= 1 ,x ++ ;
  vector<FFT::Base> a,b;a.resize ( sz , FFT::Base( 0 ) ) , b . resize ( sz ,
   FFT::Base( 0 ) );
  for(int i = 0;i<n;i++) a[i] = FFT::Base ( v0[i] );
  for(int i = 0;i<m;i++) b[i] = FFT::Base ( v1[i] ) ;
  for(auto w : a ) cout << w << " " ; cout << endl;
  for( auto w : b ) cout << w << " " ; cout << endl;
  FFT::rec_fft ( a , false ) , FFT::rec_fft ( b , false );
  for ( int i = 0;i<sz;i++ ) a[i] *= b[i];
  for( auto w : a ) cout << w << " " ; cout << endl;
  FFT::rec_fft ( a , true );
```

```
  for( auto x : a ) cout << x << " " ; cout << endl;
  return 0;
}


// Header files, namespaces,
// macros as defined above
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;

//#define ordered_set tree<int, null_type,less<int>,
 rb_tree_tag,tree_order_statistics_node_update>
typedef tree<int, null_type,less<int>,
 rb_tree_tag,tree_order_statistics_node_update> ordered_set;


#include <ext/rope> //header with rope
using namespace std;
using namespace __gnu_cxx; //namespace with rope and some additional stuff
rope <int> v; //use as usual STL container
// arbiterary insert, erase, substring
rope <int> cur = v.substr(l, r - l + 1);
v.erase(l, r - l + 1);
v.insert(v.mutable_begin(), cur);
v.push_back(i); //initialization
```