#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <stdbool.h>

// Function to compute (a^b) % mod using fast exponentiation

int mod\_exp(int base, int exp, int mod) {

int result = 1;

while (exp > 0) {

if (exp % 2 == 1) {

result = (result \* base) % mod;

}

base = (base \* base) % mod;

exp /= 2;

}

return result;

}

// Function to check if a number is primitive in F\_p

bool is\_primitive(int g, int p) {

for (int i = 1; i < p - 1; i++) {

if (mod\_exp(g, i, p) == 1) {

return false;

}

}

return true;

}

// Find a primitive element in F\_p

int find\_primitive(int p) {

for (int g = 2; g < p; g++) {

if (is\_primitive(g, p)) {

return g;

}

}

return -1;

}

// Polynomial multiplication modulo an irreducible polynomial

int poly\_mult\_mod(int a, int b, int mod\_poly, int p) {

int result = 0;

while (b) {

if (b & 1) {

result ^= a;

}

b >>= 1;

a <<= 1;

if (a & (1 << 4)) { // Assuming degree 4 field extension

a ^= mod\_poly;

}

}

return result;

}

// Find a primitive element in F\_{p^m}

bool is\_primitive\_ext(int g, int mod\_poly, int p, int field\_size) {

int value = g;

for (int i = 1; i < field\_size - 1; i++) {

value = poly\_mult\_mod(value, g, mod\_poly, p);

if (value == 1) return false;

}

return true;

}

int find\_primitive\_ext(int mod\_poly, int p, int field\_size) {

for (int g = 2; g < field\_size; g++) {

if (is\_primitive\_ext(g, mod\_poly, p, field\_size)) {

return g;

}

}

return -1;

}

// Check if a polynomial is irreducible in F\_2[x]

bool is\_irreducible(int poly, int degree) {

for (int div = 2; div < (1 << (degree / 2 + 1)); div++) {

int remainder = poly;

int divisor = div;

while (remainder >= divisor) {

int shift = (int)log2(remainder) - (int)log2(divisor);

remainder ^= (divisor << shift);

}

if (remainder == 0) {

return false;

}

}

return true;

}

// Find an irreducible monic polynomial of given degree over F\_2

int find\_irreducible\_monic(int degree) {

for (int poly = (1 << degree) | 1; poly < (1 << (degree + 1)); poly += 2) {

if (is\_irreducible(poly, degree)) {

return poly;

}

}

return -1;

}

int main() {

int p = 7; // Prime field F\_p

printf("Primitive element in F\_%d: %d\n", p, find\_primitive(p));

int mod\_poly = 0b10011; // x^4 + x + 1 (irreducible over F\_2)

int field\_size = 16; // 2^4

printf("Primitive element in F\_%d^4: %d\n", p, find\_primitive\_ext(mod\_poly, 2, field\_size));

int degree = 4;

int irreducible\_poly = find\_irreducible\_monic(degree);

printf("Irreducible monic polynomial of degree %d over F\_2: %04b\n", degree, irreducible\_poly);

return 0;

}