```
#include <stdio.h>
#include <stdbool.h>

#define MAX_M 10 // Maximum field size for practical display

// Check if a polynomial is irreducible in GF(2)
bool is_irreducible(int poly, int m) {
    int x = 2; // x in binary (10)
    int mod = (1 << m) | 1; // Generate a polynomial of degree m

    for (int i = 1; i < (1 << m) - 1; i++) {
        x = (x << 1) ^ ((x & (1 << (m - 1))) ? poly : 0);
        if (x == 2) return false; // If we loop back to x, it's reducible
    }
    return true;
}

// Find the first irreducible polynomial of degree m
int find_irreducible_polynomial(int m) {
    for (int poly = (1 << m) | 1; poly < (1 << (m + 1)); poly += 2) {
        if (is_irreducible(poly, m)) {
            return poly;
        }
    }
    return -1;
}

// Generate field elements in GF(2^m)
void generate_field_elements(int *elements, int mod_poly, int field_size) {
    elements[0] = 1;
    for (int i = 1; i < field_size - 1; i++) {
        elements[i] = elements[i - 1] << 1;
        if (elements[i] & field_size) {
            elements[i] ^= mod_poly;
        }
    }
}

// Print field elements in GF(2^m)
void print_field_elements(int *elements, int field_size) {
    printf("Elements of GF(2^m):\n");
    printf("0 "); // Zero element
    for (int i = 0; i < field_size - 1; i++) {
        printf("%d ", elements[i]);
    }
    printf("\n");
}

int main() {
    int m;
    printf("Enter m for GF(2^m): ");
    scanf("%d", &m);

    if (m > MAX_M) {
        printf("m is too large for display purposes!\n");
        return 1;
    }

    int field_size = 1 << m; // 2^m elements
    int mod_poly = find_irreducible_polynomial(m);

    if (mod_poly == -1) {
        printf("No irreducible polynomial found for m = %d!\n", m);
        return 1;
    }

    int elements[field_size - 1];
    generate_field_elements(elements, mod_poly, field_size);
    print_field_elements(elements, field_size);

    return 0;
}
```