```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

long gcd(long a, long b) {
    while (b != 0) {
        long t = b;
        b = a % b;
        a = t;
    }
    return a;
}

long mod_exp(long base, long exp, long mod) {
    long result = 1;
    base = base % mod;
    while (exp > 0) {
        if (exp % 2 == 1) {
            result = (result * base) % mod;
        }
        exp = exp >> 1;
        base = (base * base) % mod;
    }
    return result;
}

long mod_inverse(long a, long m) {
    long m0 = m, t, q;
    long x0 = 0, x1 = 1;
    if (m == 1) return 0;
    while (a > 1) {
        q = a / m;
        t = m;
        m = a % m;
        a = t;
        t = x0;
        x0 = x1 - q * x0;
        x1 = t;
    }
    if (x1 < 0) x1 += m0;
    return x1;
}

void rsa_generate_keys(long *e, long *d, long *n) {
    long p = 61, q = 53;
    *n = p * q;
    long phi = (p - 1) * (q - 1);
    *e = 17;
    while (gcd(*e, phi) != 1) {
        (*e)++;
    }
    *d = mod_inverse(*e, phi);
}

long rsa_encrypt(long msg, long e, long n) {
```

```
    return mod_exp(msg, e, n);
}

long rsa_decrypt(long cipher, long d, long n) {
    return mod_exp(cipher, d, n);
}

int main() {
    long e, d, n;
    rsa_generate_keys(&e, &d, &n);

    long msg = 65;
    printf("Original message: %ld\n", msg);

    long cipher = rsa_encrypt(msg, e, n);
    printf("Encrypted message: %ld\n", cipher);

    long decrypted = rsa_decrypt(cipher, d, n);
    printf("Decrypted message: %ld\n", decrypted);

    return 0;
}
```