```c
#include <stdio.h>

// Function for extended Euclidean Algorithm
int extendedGcd(int a, int b, int *x, int *y) {
    // Base Case
    if (a == 0) {
        *x = 0;
        *y = 1;
        return b;
    }

    int x1, y1; // To store results of recursive call
    int gcd = extendedGcd(b % a, a, &x1, &y1);

    // Update x and y using results of recursive call
    *x = y1 - (b / a) * x1;
    *y = x1;

    return gcd;
}

// Function to find modular multiplicative inverse
void modInverse(int A, int M) {
    int x, y;
    int g = extendedGcd(A, M, &x, &y);

    if (g != 1) {
        printf("Inverse doesn't exist (A and M are not coprime)\n");
    } else {
        // m is added to handle negative x
        int res = (x % M + M) % M;
        printf("Modular multiplicative inverse of %d modulo %d is %d\n", A, M, res);
    }
}

// Driver Code
int main() {
    int A = 3, M = 11;
    modInverse(A, M);

    A = 10, M = 17;
    modInverse(A, M);

    A = 4, M = 6; // Example where inverse doesn't exist
    modInverse(A, M);

    return 0;
}

// OUTPUT
// Modular multiplicative inverse of 3 modulo 11 is 4
// Modular multiplicative inverse of 10 modulo 17 is 12
// Inverse doesn't exist (A and M are not coprime)
```