// C++ program to do modular division

#include<iostream>

using namespace std;

// C function for extended Euclidean Algorithm

int gcdExtended(int a, int b, int \*x, int \*y);

// Function to find modulo inverse of b. It returns

// -1 when inverse doesn't

int modInverse(int b, int m)

{

    int x, y; // used in extended GCD algorithm

    int g = gcdExtended(b, m, &x, &y);

    // Return -1 if b and m are not co-prime

    if (g != 1)

        return -1;

    // m is added to handle negative x

    return (x%m + m) % m;

}

// Function to compute a/b under modlo m

void modDivide(int a, int b, int m)

{

    a = a % m;

    int inv = modInverse(b, m);

    if (inv == -1)

       cout << "Division not defined";

    else

       cout << "Result of division is " << (inv \* a) % m;

}

// C function for extended Euclidean Algorithm (used to

// find modular inverse.

int gcdExtended(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 = gcdExtended(b%a, a, &x1, &y1);

    // Update x and y using results of recursive

    // call

    \*x = y1 - (b/a) \* x1;

    \*y = x1;

    return gcd;

}