**Cryptography, Network and Security**

Assignment 4

4. Implementation of Chinese Remainder Theorem (CRT)

Code:

#include <iostream>

#include <vector>

using namespace std;

*// Function to compute the GCD and the coefficients x and y for the equation ax + by = gcd(a, b)*

int extendedEuclidean(int a, int b, int *&*x, int *&*y) {

    if (b == 0) {

        x = 1;

        y = 0;

        return a;

    }

    int x1, y1;

    int gcd = extendedEuclidean(b, a % b, x1, y1);

    x = y1;

    y = x1 - (a / b) \* y1;

    return gcd;

}

*// Function to find modular inverse using Extended Euclidean Algorithm*

int modInverse(int a, int m) {

    int x, y;

    int g = extendedEuclidean(a, m, x, y);

    if (g != 1) {

        cout << "Inverse doesn't exist!";

        return -1;

    } else {

        return (x % m + m) % m;

    }

}

*// Function to solve the system of congruences using the Chinese Remainder Theorem*

int chineseRemainder(vector<int> num, vector<int> rem, int n) {

    int prod = 1;

    for (int i = 0; i < n; i++) {

        prod \*= num[i];

    }

    int result = 0;

    for (int i = 0; i < n; i++) {

        int pp = prod / num[i];

        int inv = modInverse(pp, num[i]);

        result += rem[i] \* inv \* pp;

    }

    return result % prod;

}

int main() {

*// System of equations:*

*// x ≡ rem[0] (mod num[0])*

*// x ≡ rem[1] (mod num[1])*

*// x ≡ rem[2] (mod num[2])*

    vector<int> num = {3, 4, 5};

    vector<int> rem = {2, 3, 1};

    int n = num.size();

    int result = chineseRemainder(num, rem, n);

    cout << "x is " << result << endl;

    return 0;

}