21CSE1016, Ketan Kunkalikar. Lab Date: 21/08/24.

// Q1. Write a program to implement the Chinese Remainder Theorem.

#include <bits/stdc++.h>

using namespace std;

long long MI\_using\_EEA(long long a, long long b)

{

    // here, a is our n from Zn. and b is the number we want to find MI for.

    long long r1 = a, r2 = b;

    long long s1 = 1, s2 = 0;

    long long t1 = 0, t2 = 1;

    while (r2 != 0)

    {

        long long q = r1 / r2;

        r1 = r1 - q \* r2;

        swap(r1, r2);

        s1 = s1 - q \* s2;

        swap(s1, s2);

        t1 = t1 - q \* t2;

        swap(t1, t2);

    }

    if (t1 < 0)

        t1 += a;

    return t1;

}

int main()

{

    long long n;

    cout << "Enter the number of equations: ";

    cin >> n;

    // intake of equations.

    vector<long long> a(n), m(n);

    for (long long i = 0; i < n; i++)

    {

        cout << "Enter coefficient " << i + 1 << ": ";

        cin >> a[i];

    }

    for (long long i = 0; i < n; i++)

    {

        cout << "Enter modulus " << i+1 << ": ";

        cin >> m[i];

    }

    // evaluating M.

    long long M = 1;

    for (long long i = 0; i < n; i++)

    {

        M \*= m[i];

    }

    long long ans = 0;

    for (long long i = 0; i < n; i++)

    {

        long long Mi = M / m[i];

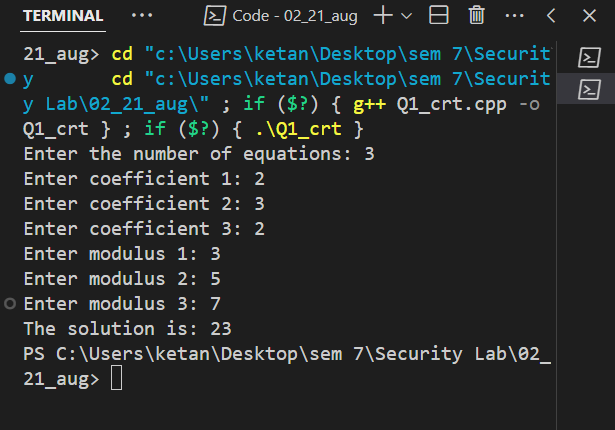
        ans += a[i] \* Mi \* MI\_using\_EEA(m[i], Mi);

    }

    cout << "The solution is: " << ans % M << "\n";

    return 0;

}



//Q2. write a c++ program to find the modular inverse of a 3x3 matrix using EEA.

#include <bits/stdc++.h>

using namespace std;

void print\_matrix(long long matrix[3][3]) {

    for (long long i = 0; i < 3; i++) {

        for (long long j = 0; j < 3; j++) {

            cout << matrix[i][j] << " ";

        }

        cout << "\n";

    }

}

long long determinant(long long matrix[3][3]) {

    long long det = 0;

    det = matrix[0][0] \* (matrix[1][1] \* matrix[2][2] - matrix[1][2] \* matrix[2][1])

        - matrix[0][1] \* (matrix[1][0] \* matrix[2][2] - matrix[1][2] \* matrix[2][0])

        + matrix[0][2] \* (matrix[1][0] \* matrix[2][1] - matrix[1][1] \* matrix[2][0]);

    return det;

}

long long MI\_using\_EEA(long long a, long long b) {

    long long r1 = a, r2 = b;

    long long s1 = 1, s2 = 0;

    long long t1 = 0, t2 = 1;

    while (r2 != 0) {

        long long q = r1 / r2;

        r1 = r1 - q \* r2;

        swap(r1, r2);

        s1 = s1 - q \* s2;

        swap(s1, s2);

        t1 = t1 - q \* t2;

        swap(t1, t2);

    }

    if (t1 < 0)

        t1 += a;

    return t1;

}

void inverse\_matrix(long long matrix[3][3], long long mod) {

    long long det = determinant(matrix);

    long long det\_inv = MI\_using\_EEA(mod, (det % mod + mod) % mod);

    long long adj[3][3];

    for (long long i = 0; i < 3; i++) {

        for (long long j = 0; j < 3; j++) {

            long long minor[2][2];

            long long m = 0, n = 0;

            for (long long k = 0; k < 3; k++) {

                for (long long l = 0; l < 3; l++) {

                    if (k != i && l != j) {

                        minor[m][n++] = matrix[k][l];

                        if (n == 2) {

                            m++;

                            n = 0;

                        }

                    }

                }

            }

            adj[j][i] = ((i + j) % 2 == 0 ? 1 : -1) \* (minor[0][0] \* minor[1][1] - minor[0][1] \* minor[1][0]);

        }

    }

    for (long long i = 0; i < 3; i++) {

        for (long long j = 0; j < 3; j++) {

            matrix[i][j] = ((adj[i][j] \* det\_inv) % mod + mod) % mod;

        }

    }

}

int main() {

    long long matrix[3][3];

    long long det = 0;

    cout << "Enter the elements of the 3x3 matrix:\n";

    for (long long i = 0; i < 3; i++) {

        for (long long j = 0; j < 3; j++) {

            cout << "Enter element at position [" << i+1 << "][" << j+1 << "]: ";

            cin >> matrix[i][j];

        }

    }

    long long mod;

    cout << "Enter the modulus: ";

    cin >> mod;

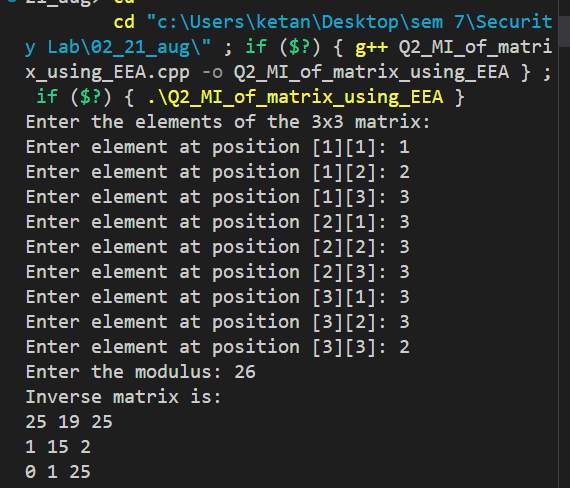
    inverse\_matrix(matrix, mod);

    cout << "Inverse matrix is: \n";

    print\_matrix(matrix);

    return 0;

}



// Q3. Write a program to implement modular exponentiation using repeated squaring and multiplication algorithm.

#include <bits/stdc++.h>

using namespace std;

long long Repeat\_Sq\_and\_mult(long long A, long long k, long long mod) {

    long long b = 1;

    while (k > 0) {

        if (k % 2 == 1) {

            b = (b \* A) % mod;

        }

        A = (A \* A) % mod;

        k /= 2;

    }

    return b;

}

int main() {

    long long A, k, mod;

    cout << "Enter the base: ";

    cin >> A;

    cout << "Enter the exponent: ";

    cin >> k;

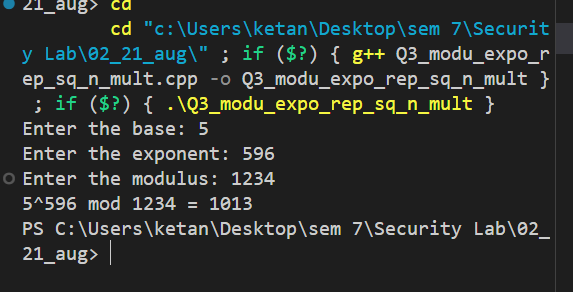
    cout << "Enter the modulus: ";

    cin >> mod;

    cout << A << "^" << k << " mod " << mod << " = " << Repeat\_Sq\_and\_mult(A, k, mod) << "\n";

    return 0;

}



// Q4. Write a program to find Euler totient value in Zn.

#include <bits/stdc++.h>

using namespace std;

long long gcd(long long a, long long b)

{

    if (a == 0)

        return b;

    return gcd(b % a, a);

}

long long phi(long long n)

{

    long long result = 1;

    for (long long i = 2; i < n; i++)

        if (gcd(i, n) == 1)

            result++;

    return result;

}

int main()

{

    long long n;

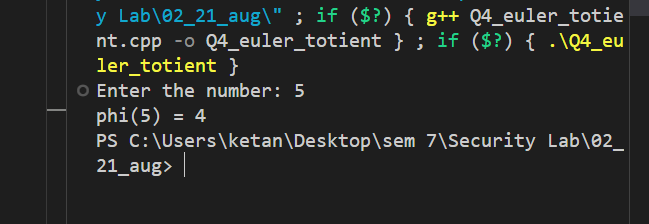
    cout << "Enter the number: ";

    cin >> n;

    cout << "phi("<<n<<") = " << phi(n) << "\n";

    return 0;

}



// Q5. Write a program to find the order of modulo in Zn.

// order of Zn

#include <bits/stdc++.h>

using namespace std;

vector<long long> calculateMI(long long n)

{

    vector<long long> mi;

    for (long long i = 1; i < n; i++)

    {

        for (long long j = 1; j < n; j++)

        {

            long long product = i \* j;

            if (product % n == 1)

            {

                mi.push\_back(i);

                cout << "mi value " << i << endl;

            }

        }

    }

    return mi;

}

vector<long long> calculateFactor(long long phi)

{

    vector<long long> phiFactors;

    for (long long i = 1; i <= phi; i++)

    {

        if (phi % i == 0)

        {

            phiFactors.push\_back(i);

            cout << "phi factors " << i << endl;

        }

    }

    return phiFactors;

}

void calculateOrder(vector<long long> phiFactors, vector<long long> mi, long long n)

{

    long long p = mi.size();

    long long q = phiFactors.size();

    for (long long i = 0; i < p; i++)

    {

        long long a = mi[i];

        for (long long j = 0; j < q; j++)

        {

            long long t = phiFactors[j];

            long long r = t % 22;

            long long ex = t / 2 + r;

            long long v1 = (pow(a, t / 2));

            long long v2 = pow(a, ex);

            long long val = (v1 % n) \* (v2 % n);

            // cout<<"val="<<val<<" "<<"t="<<t<<ends<<"a="<<a<<endl;

            if (val % n == 1)

            {

                cout << "val=" << val << endl;

                cout << "t(" << a << ")=" << t << endl;

                break;

            }

        }

    }

}

int main()

{

    long long n;

    cout << "enter n value in Zn" << endl;

    cin >> n;

    vector<long long> mi; // Zn\*

    mi = calculateMI(n);

    long long phi = mi.size();

    cout << "phi=" << phi << endl;

    vector<long long> phiFactors;

    phiFactors = calculateFactor(phi);

    calculateOrder(phiFactors, mi, n);

    return 0;

}

