Name: Rutikesh Sawant

Batch: B2

Subject: CNS Lab

PRN: 2019BTECS00034

Assignment 13

Aim: Chinese Remainder Theorem implementation

Theory:

x = a1 (mod n1)

...

x = ak (mod nk)

This is equivalent to saying that x mod ni = ai (for i=1…k). The notation above is common in group theory, where you can define the group of integers modulo some number n and then you state equivalences (or congruence) within that group.

So x is the unknown; instead of knowing x, we know the remainder of the division of x by a group of numbers. If the numbers ni are pairwise coprimes (i.e. each one is coprime with all the others) then the equations have exactly one solution. Such solution will be modulo N, with N equal to the product of all the n_i.

Code:

```
#include <bits/stdc++.h>
using namespace std;

void file()
```

```
{
#ifndef ONLINE_JUDGE
	freopen("input.txt", "r", stdin);
	freopen("output.txt", "w", stdout);
#endif
}

// Function for extended Euclidean Algorithm
int ansS, ansT;
int findGcdExtended(int r1, int r2, int s1, int s2, int t1, int t2)
{
	// Base Case
	if (r2 == 0)
	{
		ansS = s1;
		ansT = t1;
		return r1;
	}

	int q = r1 / r2;
	int r = r1 % r2;

	int s = s1 - q * s2;
	int t = t1 - q * t2;

	cout << q << " " << r1 << " " << r2 << " " << r << " " << s1 << " " << s2 << " " << s
<< " " << t1 << " " << t2 << " " << t << endl;
```

```
	return findGcdExtended(r2, r, s2, s, t2, t);
}


int modInverse(int A, int M)
{
	int x, y;
	int g = findGcdExtended(A, M, 1, 0, 0, 1);
	if (g != 1) {
		cout << "Inverse doesn't exist";
		return 0;
	}
	else {

		// m is added to handle negative x


		int res = (ansS % M + M) % M;
		cout << "inverse is " << res << endl;
		return res;
	}
}


int findX(vector<int> num, vector<int> rem, int k)
{
	// Compute product of all numbers
```

```
    int prod = 1;
    for (int i = 0; i < k; i++)
        prod *= num[i];

    // Initialize result
    int result = 0;

    // Apply above formula
    for (int i = 0; i < k; i++) {
        int pp = prod / num[i];
        result += rem[i] * modInverse(pp, num[i]) * pp;
    }

    return result % prod;
}



int main()
{
    file();

    // 3
    // 3 4 5
    // 2 3 1

    int k;
    cin >> k;
```

```
    vector<int> num(k), rem(k);
    for (int i = 0; i < k; i++)
        cin >> num[i];
    for (int i = 0; i < k; i++)
        cin >> rem[i];

    int x = findX(num, rem, k);
    cout << "x is " << x;
    return 0;
}
```

Output:

```
6 20 3 2 1 0 1 0 1 -6
1 3 2 1 0 1 -1 1 -6 7
2 2 1 0 1 -1 3 -6 7 -20
inverse is 2
3 15 4 3 1 0 1 0 1 -3
1 4 3 1 0 1 -1 1 -3 4
3 3 1 0 1 -1 4 -3 4 -15
inverse is 3
2 12 5 2 1 0 1 0 1 -2
2 5 2 1 0 1 -2 1 -2 5
2 2 1 0 1 -2 5 -2 5 -12
inverse is 3
x is 11
```