C&NS Lab Assignment 11

Onkar Santosh Gavali (2019BTECS00037)

Batch B2

# Index

Chinese Reminder Theorem

* Explain the Chinese Reminder Theorem
* Implement the Chinese Reminder Theorem using any programming language.

# 

# 

# Chinese Reminder Theorem

Chinese Reminder Theorem used to find x in multiple congruent eqtions

X = r1 mod a1

X = r2 mod a2

X = r3 mod a3

X = r4 mod a4

# Code

#include <bits/stdc++.h>

#define ll long long

#define ul unsigned long long

#define pb emplace\_back

#define po pop\_back

#define vi vector<ll>

#define vii vector<vector<ll>>

using namespace std;

const int MODVALUE = 1e9;

long long gcdExtended(long long a, long long b, long long \*x, long long \*y)

{

cout << "a= "<<a<< " b= "<< b << " | t1= " << \*x << " t2= " << \*y << "\n";

// Base Case

if (b == 0)

{

return \*x;

}

long long q = a / b;

long long y1 = \*x - q \* (\*y);

\*x = \*y;

\*y=y1;

long long gcd = gcdExtended(b, a % b, x, y);

return gcd;

}

int main() {

char patternChar = '-';

char resetChar = ' ';

int lineWidth = 90;

int initialWidth = 50;

cout<<"demo: \n 3 \n 3 5 7 \n 2 3 2\n";

cout << setfill(patternChar) << setw(lineWidth) << patternChar << endl;

cout << setfill(resetChar);

cout << setw(initialWidth) << "Chinese Remainder Theorm" << endl;

cout << setfill(patternChar) << setw(lineWidth) << patternChar << endl;

cout << setfill(resetChar);

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

int n;

cin >> n;

vector<int> divisor(n, 0);

vector<int> remainder(n, 0);

// M = m1 \* m2 \* m3 \* .....

long long int M = 1;

cout << "Enter the divisors of " << n << " the equations: " ;

for(int i = 0; i < n; i++){

cin >> divisor[i];

M \*= divisor[i];

M %= MODVALUE;

}

cout << "Enter the remainders of " << n << " equations: " ;

for(int i = 0; i < n; i++){

cin >> remainder[i];

}

// finding m1, m2, m3, ...

vector<int> mValues(n);

vector<int> invMValues(n);

for(int i = 0; i < n; i++){

mValues[i] = M/divisor[i];

long long x=0, y=1;

x = gcdExtended(divisor[i],mValues[i],&x, &y);

cout<<"The inverse for M"<<(i+1)<<" = "<<mValues[i]<<" is "<<x<<"\n";

invMValues[i] = x;

}

long long ans = 0;

for(int i = 0; i < n; i++){

ans += (((1LL\* remainder[i] \* mValues[i])%M)\*invMValues[i])%M;

ans %= M;

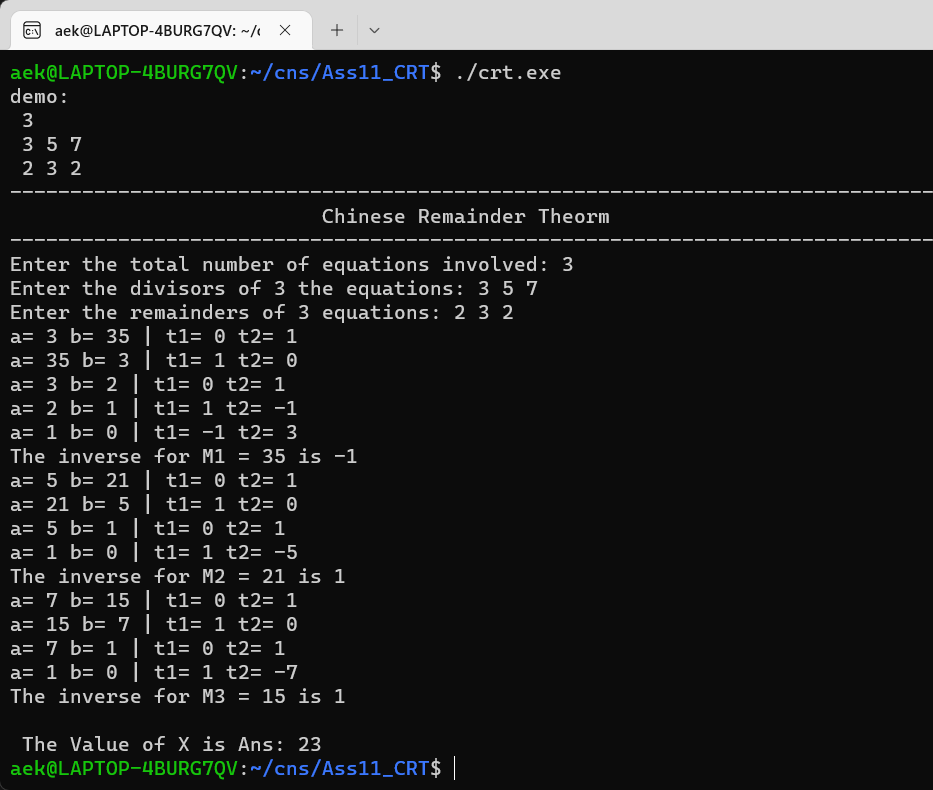
}

cout << "\n The Value of X is Ans: " << ans << endl;

return 0;

}

# Output



# 