Maitreya Kanitkar

BE-IT 8084

# ICS Assignment 1: Chinese remainder theorem

---

**Title:** Develop and program in C++ based on number theory such as Chinese remainder.

**Objective:**
Understand and implement the Chinese remainder theorem.

**Theory:**
*Chinese Remainder Theorem states that there always exists an x that satisfies given congruences.*Let num[0], num[1], …num[k-1] be positive integers that are pairwise coprime. Then, for any given sequence of integers rem[0], rem[1], … rem[k-1], there exists an integer x solving the following system of simultaneous congruences.
The first part is clear that there exists an x. The second part basically states that all solutions (including the minimum one) produce the same remainder when divided by-product of n[0], num[1], .. num[k-1]. In the above example, the product is 3*4*5 = 60. And 11 is one solution, other solutions are 71, 131, .. etc. All these solutions produce the same remainder when divided by 60, i.e., they are of form 11 + m*60 where m >= 0.
A Naive Approach to find x is to start with 1 and one by one increment it and check if dividing it with given elements in num[] produces corresponding remainders in rem[]. Once we find such an x, we return it.
Below is the implementation of the Naive Approach.
*Time Complexity:* O(M), M is the product of all elements of num[] array.
*Auxiliary Space:* O(1)

**Algorithm:**

1. Take divisor and subsequent remainders as input from the user.
2. Check if the remainders are relatively prime or not.
3. Calculate Ni for all divisors.
4. Calculate inverse.
5. Calculate the final output and print it.

**Program:**

```
#include<iostream>
#include<vector>
using namespace std;

class crt
{
    vector<int>bi, ni, Ni, yi;
    int n, N=1, x=0;

    public:

    int pipeline()
    {
        input();
        if(relativePrime()==0)
        return 0;
        calcNi();
        inverse();
        return output();
    }

    void input()
    {
        int r, d;

        cout<<"Enter the number of divisiors or remainders : ";
        cin>>n;

        cout<<endl<<"Enter the divisors and the corresponding remainders : "<<endl;
            for(int i=0; i<n; i++)
        {
                    cin>>d>>r;
```

```
                        ni.push_back(d);
        bi.push_back(r);
            }
}

int gcd(int num1, int num2)
{
    if (num2==0)
    return num1;
    return gcd(num2, num1%num2);
}

int relativePrime()
{
    for(int i=0; i<n-1; i++)
                    for(int j=i+1; j<n; j++)
                    if(gcd(ni.at(i), ni.at(j))!=1)
                    return 0;
            return 1;
}

void calcNi()
{
    for(int i=0;i<n;i++)
    N*=ni.at(i);

    for(int i=0;i<n;i++)
    Ni.push_back(N/ni.at(i));
}

void inverse()
{
    for(int i=0;i<n;i++)
                    for(int j=1;j<ni.at(i);j++)
```

```
                    if((Ni[i]*j)%ni[i]==1)
        {
                            yi.push_back(j);
                            break;
                    }
    }

    int output()
    {
        for(int i=0;i<n;i++)
        x+=bi.at(i)*Ni.at(i)*yi.at(i);
        return x%N;
    }
};

int main()
{
    int solution;
    crt obj;

    solution=obj.pipeline();

    if(solution==0)
    cout<<"Solution not possible as divisors aren't relatively prime.";
    else
    cout<<endl<<"X="<<solution;

    return 0;
}
```

/*

OUTPUT 1:

Enter the number of divisors or remainders : 3

Enter the divisors and the corresponding remainders :

3 2

4 3

5 1

X=11

OUTPUT 2:

Enter the number of divisors or remainders : 3

Enter the divisors and the corresponding remainders :

3 2

3 4

5 1

The solution is not possible as divisors aren't relatively prime.