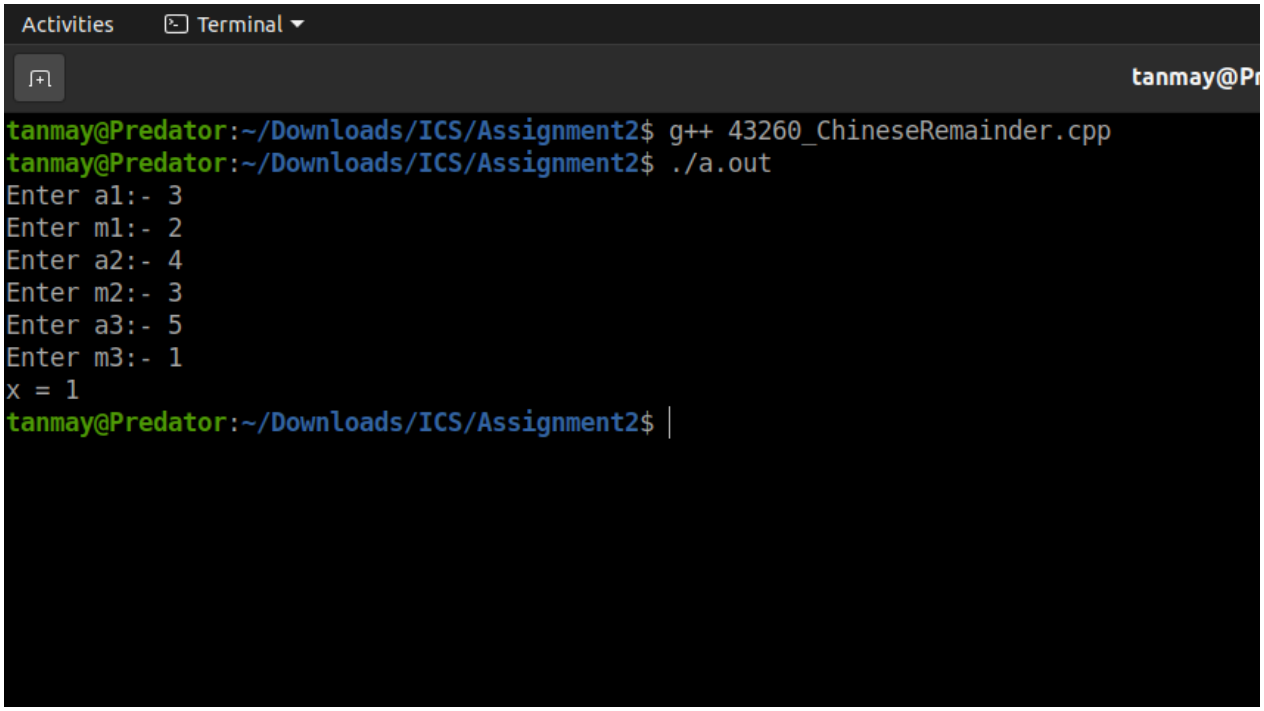
Chinese_Remainder.cpp:

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

int gcd(int a, int b) {
   if (b == 0) {
       return a;
   }
   if (a < b) {
       return gcd(b, a);
   }
   return gcd(b, a % b);
}

bool isRelativelyPrime(int m[]) {
   if(gcd(m[0], m[1]) == 1) {
       if (gcd(m[1], m[2]) == 1) {
           return gcd(m[0], m[2]) == 1;
       }
   }
   return false;
}

int inverse(int a, int m) {
   int m0 = m, t, q;
   int x0 = 0, x1 = 1;

   if (m == 1)
       return 0;

   while (a > 1) {
       q = a / m;
       t = m;
       m = a % m;
       a = t;
       t = x0;
       x0 = x1 - q * x0;
       x1 = t;
   }
```

```
    if (x1 < 0)
        x1 += m0;

    return x1;
}

int main()
{
    int a[3], m[3];

    for (int i = 0; i < 3; i++) {
        cout<<"Enter a"<<i+1<<":- ";
        cin>>a[i];
        cout<<"Enter m"<<i+1<<":- ";
        cin>>m[i];
    }

    if (isRelativelyPrime(m)) {
        int M = m[0] * m[1] * m[2];

        int M1 = M/m[0];
        int M2 = M/m[1];
        int M3 = M/m[2];

        int M1i = inverse(M1, m[0]);
        int M2i = inverse(M2, m[1]);
        int M3i = inverse(M3, m[2]);

        int temp = (a[0]*M1*M1i + a[1]*M2*M2i + a[2]*M3*M3i);

        int result = temp % M;

        cout<<"x = "<<result<<endl;
    }

    return 0;
}
```

Output:



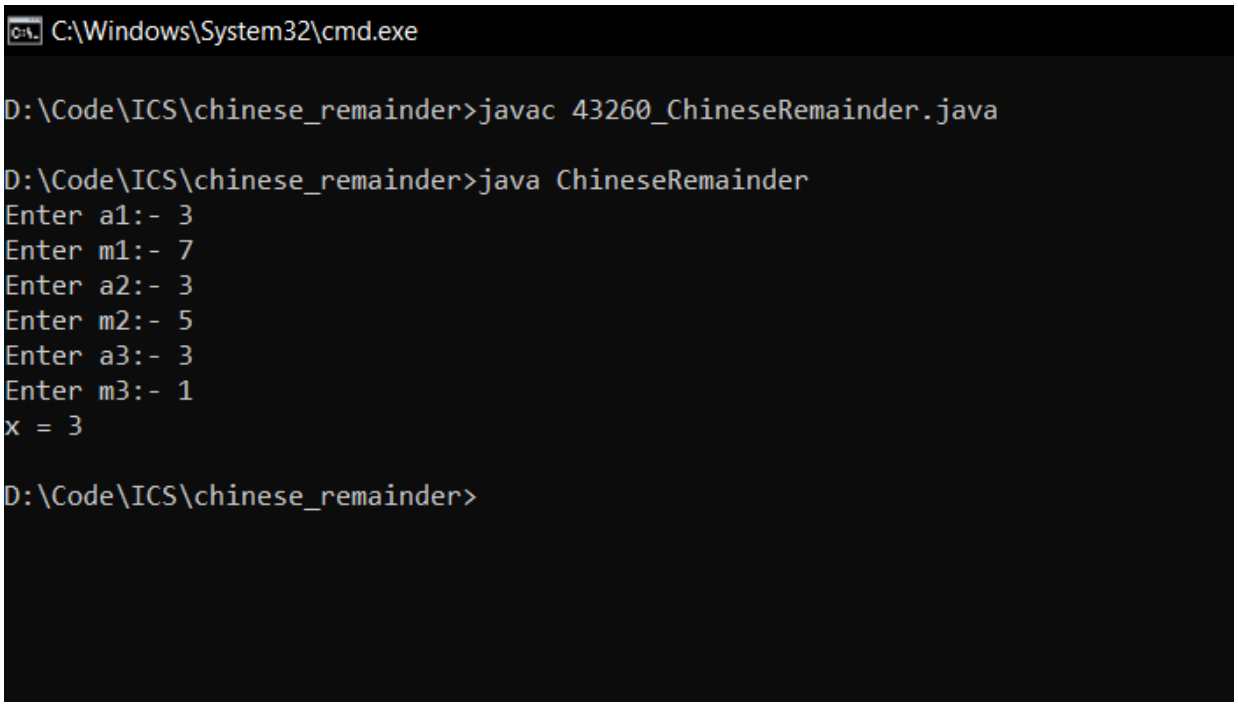
Chinese_Remainder.java:

```java
import java.util.Scanner;

class ChineseRemainder {
   public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       int[] a = new int[3];
       int[] m = new int[3];

       for (int i = 0; i < 3; i++) {
           System.out.print("Enter a" + (i + 1) + ":- ");
           a[i] = sc.nextInt();
           System.out.print("Enter m" + (i + 1) + ":- ");
           m[i] = sc.nextInt();
       }
```

```
        if (Util.isRelativelyPrime(m)) {
            int M = m[0] * m[1] * m[2];

            int M1 = M/m[0];
            int M2 = M/m[1];
            int M3 = M/m[2];

            int M1i = Util.inverse(M1, m[0]);
            int M2i = Util.inverse(M2, m[1]);
            int M3i = Util.inverse(M3, m[2]);

            int temp = (a[0]*M1*M1i + a[1]*M2*M2i + a[2]*M3*M3i);

            int result = temp % M;

            System.out.println("x = " + result);
        }
    }
}
class Util {

    static int gcd(int a, int b) {
        if (b == 0) {
            return a;
        }
        if (a < b) {
            return gcd(b, a);
        }
        return gcd(b, a % b);
    }

    static boolean isRelativelyPrime(int []m) {
        if(gcd(m[0], m[1]) == 1) {
            if (gcd(m[1], m[2]) == 1) {
                return gcd(m[0], m[2]) == 1;
            }
        }
        return false;
    }
```

```
    static int inverse(int a, int m) {
        int m0 = m, t, q;
        int x0 = 0, x1 = 1;

        if (m == 1)
            return 0;

        while (a > 1) {
            q = a / m;
            t = m;
            m = a % m;
            a = t;
            t = x0;
            x0 = x1 - q * x0;
            x1 = t;
        }
        if (x1 < 0)
            x1 += m0;

        return x1;
    }
}
```

Output:


C:\Windows\System32\cmd.exe
D:\Code\ICS\chinese_remainder>javac 43260_ChineseRemainder.java
D:\Code\ICS\chinese_remainder>java ChineseRemainder
Enter a1:- 3
Enter m1:- 7
Enter a2:- 3
Enter m2:- 5
Enter a3:- 3
Enter m3:- 1
x = 3
D:\Code\ICS\chinese_remainder>