CMSC 2123 Discrete Structures

Project 4

Due: See the due date on D2L.

1. Basic Requirements

In this project, you need to implement a toolkit to serve the following functionalities:

(1). Calculate the greatest common divisor of two integers (say a and b) using the Euclidean Algorithm: get $gcd(a, b)$. Both a and b should be positive integers from command line input.

(2). Given an integer a and the modulus m, get the inverse of a mod m (the result should be positive and smaller than m). Both a and m should be positive integers from command line input, and $a < m$, $gcd(a, m) = 1$.

(3). Solve a linear congruence which is from the command line input. Let's say your input is $ax \equiv b \ (mod\ m)$, a, b and m should be positive integers, $a < m$, $b < m$ and $gcd(a, m) = 1$. If the result is $x \equiv c \ (mod\ m)$, then c should be a positive integer and $c < m$.

You will need the following algorithms:

(a) The pseudocode for the Euclidean algorithm:

```
procedure gcd(a, b: positive integers)
    x := a
    y := b
        while   y ≠ 0
            r := x mod y
            x := y
            y := r
    return x    {gcd(a,b) is x}
```

(b) The pseudocode for the Extended Euclidean algorithm:

```
procedure extendedEuclidean(a, b: positive integers)
    x := a
    y := b
    oldolds := 1
    olds := 0
    oldoldt := 0
    oldt := 1
    while y ≠ 0
        q := x div y
        r := x mod y
        x := y
        y := r
        s := oldolds - q * olds
        t := oldoldt - q * oldt
        oldolds := olds
        oldoldt := oldt
        olds := s
        oldt := t     {gcd(a, b) is x, and oldolds*a + oldoldt*b = x}
```

From the pseudocode of the Extended Euclidean algorithm, you can tell it can be used to calculate the inverse of a mod m. You need to make necessary changes to work with our project.

Once you can find the inverse of a mod m, then the linear congruence can be solved easily.

You need to implement the entire project, including parsing the command line and implementation of three functionalities. For each functionality, you need to check whether the input values are valid, e.g. check $a < m$, $gcd(a, m) = 1$. These restrictions are described as above, so that the algorithms can be correctly applied.

2. Program Structure

2.1 Program Files

Project **4** consists of the file **p04.cpp.** The file name for project 4 is exactly as given.

2.2 Command Line Input:

The general syntax for command line input is

$ ./p04 functionality_number [parameters]

The functionality_number is used as the index for each functionality, i.e., 1 for calculating the greatest common divisor of two integers (say a and b) using the Euclidean Algorithm; 2 for calculating the inverse of a mod m; 3 for solving the linear congruence $ax \equiv b \ (mod\ m)$.

functionality_number	functionality
1	Finding the greatest common divisor
2	Finding the inverse
3	Solving the linear congruence

The [parameters] are the numbers or expression needed for that functionality. For example:

(1). If you want to find the greatest common divisor of 91 and 287, you should give the following command:

$ ./p04 1 91 287

(2). If you want to find the inverse of 3 mod 7, you should give the following command:

$ ./p04 2 3 7

(3). if you want to solve the linear congruence $3x \equiv 4\ (mod\ 7)$, you should give the following command:

$ ./p04 3 3 4 7

3, 4 and 7 can be parsed for the expression of $3x \equiv 4\ (mod\ 7)$. Once it's solved you need to display this linear congruence on the screen, you can display it as **3x===4(mod7)**. There is no space in this expression.

2.3 Output on the Screen:

Just display the result as following:

(1). For finding the greatest common divisor:

$ ./p04 1 91 287

gcd(91, 287) = 7

(2). For finding the inverse:

$ ./p04 2 3 7

5 is the inverse of 3 modulo 7

(3). For solving the linear congruence:

$ ./p04 3 3 4 7

The solution for 3x===4(mod7) is x===6(mod7)

3. Submission

Please make your submission on D2L. This report should include a copy of your source code, and the result of a sample execution. Please also include your name(s) and email(s) in the header of this report.

4. Evaluation

This project will be evaluated according to the correctness of the various tasks specified above, and secondarily according to the quality of your code. The rubric is as follows:

Categories	Weights
`Parse input parameters`	15%
`Task 1`	15%
`Task 2`	30%
`Task 3`	20%
`Correctness`	10%
`Report`	10%:

Notes:

1. To be considered on time, the program must be turned in before the due date.
2. Programs must reflect your knowledge and work, not others.
3. No points, zero (0), will be given to any program containing compilation errors.