Coding Assignment 2
ESO207 2024-25-I

August 15, 2024

1 Introduction

In this homework, you will implement a C program to implement two arithmetic algorithms that were discussed in the class: modular exponentiation and extended Euclid's algorithm. The total marks will be 180.

The deadline of this homework is Friday, August 23, 11 PM IST. Your submissions will not be accepted after this time. Submit a single C file titled 'hw.c' to the Gradescope active assignment titled 'arithmetic'.

2 Problem Statement

2.1 Modular Expoentiation

The input instance for this will consist of two lines. The first line will hold an option number: 0 means modular exponentiation and 1 means extended Euclid's algorithm. The next line contains three positive integers A, B, C each separated by a space. All three numbers are promised to be at most 1000. The output will consist of a single number D, no other character after it. You are supposed to output $D = A^B \ mod \ C$.

An example input is given below.

```
0
54 214 153
```

The sample output will be as follows.

```
117
```

Note that there is no character after 117.

2.2 Finding Inverse Modulo using Euclid's Algorithm

Recall that in the class we showed that the extended Euclid's algorithm returns two numbers a, b such that

$$ax + by = gcd(x, y).$$

Now, if a, b are coprimes, we get $ax + by = 1$ and hence $ax \ mod \ y = 1$. Such an a is called as the inverse of x modulo y, denoted $a = x^{-1} \ mod \ y$. $x^{-1} mody$ may not always exist, for example when $x = 10, y = 2$, ax is always even and therefore $ax \ mod \ 2$ can never be 1. However, whenever $gcd(x, y) = 1$ it can be seen that such an inverse always exists.

Once we find an inverse a it is easy to see that any $a + ty$ is also an inverse for any integer t. Therefore, given any inverse a, the number $b = a \ mod \ y$ is an integer between 0 and $y - 1$. No other number in this range apart from b will be an inverse of x. Therefore, such a b will be called the smallest inverse of x modulo y.

After obtaining the two numbers a, b from the extended Euclid's algorithm if you simply take $a \ mod \ y$ you'll obtain this smallest inverse modulo y. Be careful that if $a < 0$ the C command "$a \ \% \ n$" simply returns you a. Thus for example, "$(-493)\%788$" will return you (-493) itself whereas you want the answer to be 295. Handle this issue by writing out a simple function that returns you the correct value of $a \ mod \ y$, lying between 0 and $(y-1)$, when $a < 0$.

For this problem, the input will consist of two lines. The first line will hold an option number: 0 means modular exponentiation and 1 means extended Euclid's algorithm. The next line will contain two integers x, y separated by a space. x, y are both promised to be at most 10000. The output will consist of two integers g, b separated by a space. g will always be equal to $gcd(x, y)$. If $g = 1$, b must be equal to the smallest inverse of x modulo y. If $g > 1$, b must be 0.

First example The input is as follows.

```
1
1512 759
```

The expected output is as follows.

```
3 0
```

Second example The input is as follows.

```
1
5711 788
```

The expected output is as follows.

```
1 295
```

3 Submission Instructions

- You must submit a C program titled 'hw.c'. Other programming languages such as C++, Python, etc. are not allowed. Your code must take the input from "stdin" and write the output to "stdout".
- Your code will be automatically graded in Gradescope on some test cases as above which will be hidden from you. Therefore, you must make sure that you understand and precisely follow the expected input-output behavior.
- Please write a single C code and name it as 'hw.c'. This is extremely important. If you violate this, your code will not pass the automatic test cases even if your code runs correctly on your local machine. Common examples of failures include:
  - if you write a C++ program that has the correct input-output behavior
  - if you write two or more different C codes or .h header codes and link them
  - or write a single correct code but name it as 'test.c'

  In any of the above cases, your code will fail. Thus, while you are perfectly allowed to develop your code in your local machine and it works correctly, your code may run into problems in Gradescope until and unless you follow the above instructions.
- Submit your code on *Gradescope* active assignment titled 'arithmetic'. Otherwise your code will not be graded. In particular, do NOT submit on hello IITK or over email. Email to us (instructor or the TAs), meet us, or start a discussion in helloIITK if you run into any issues.

4 Test Cases

We have already given two three test cases in the above. We are giving 3 more test cases here.

Input.

```
0
999 123 579
```

Output.

```
222
```

Input.

```
1
9991 5791
```

Output.

```
1 950
```

Input.

```
1
1172 5912
```

Output.

```
4 0
```

All the above 6 test cases will be visible and will carry 20 marks each. There will be 3 more hidden test cases, each of 20 marks. The total marks will be 180.