Fall 2025 CIS 3362 Homework #5: Number Theory Solutions

1) (5 pts) Without the aid of a computer program, determine the prime factorization of 3,548,318,400. Show your work. You may do division on a calculator. Stating which numbers divided in evenly how many times.

<u>Solution</u>

```
3,548,318,400 / 2 = 1,774,159,200
1,774,159,200 / 2 = 887,079,600
  887,079,600 / 2 = 443,539,800
  443,539,800 / 2 = 221,769,900
  221,769,900 / 2 = 110,884,950
  110,884,950 / 2 = 55,442,475
   55,442,475 / 3 = 18,480,825
   18,480,825 / 3 = 6,160,275
    6,160,275 / 3 = 2,053,425
    2,053,425 / 3 = 684,475
      684,475 / 5 = 136,895
      136,895 / 5 = 27,379
      27,379 / 11 = 2,489
       2,489 / 19 = 131
        131 / 131 = 1
```

3,548,318,400 = **2^6 * 3^4 * 5^2 * 11^1 * 19^1 * 131^1**

2) (5 pts) What is φ(3,548,318,400)? You can use a calculator, but please show your work.

<u>Solution</u>

```
φ(3,548,318,400) = φ(2^6 * 3^4 * 5^2 * 11^1 * 19^1 * 131^1)
                 = φ(2^6) * φ(3^4) * φ(5^2) * φ(11^1) * φ(19^1) *
φ(131^1)
                 = (2^6-2^5) * (3^4-3^3) * (5^2-5^1) * (11^1-11^0) *
(19^1-19^0) * (131^1-131^0)
                 = 32 * 54 * 20 * 10 * 18 * 130
                 = 808,704,000
```

3) (5 pts) Use Fermat's Theorem to calculate the remainder when 123^{12561} is divided by 967?

Solution

```
Since GCD(123, 967) = 1, by Fermat's Theorem, 123^966 = 1 (mod 967).

123^12561 = 123^(966*13+3) (mod 967)
          = (123^966)^13 * 123^3 (mod 967)
          = 1^13 * 123^3 (mod 967)
          = 359 (mod 967)
```

4) (5 pts) Use Euler's Theorem to calculate the remainder when $29^{4286522}$ is divided by 1766107?

Solution

```
1766107 / 7 = 252301
 252301 / 7 = 36043
  36043 / 7 = 5149
  5149 / 19 = 271
  271 / 271 = 1

271 = 7^3 * 19^1 * 271^1

φ(1766107) = φ(7^3 * 19^1 * 271^1)
           = φ(7^3) * φ(19^1) * φ(271^1)
           = (7^3-7^2) * (19^1-19^0) * (271^1-271^0)
           = 294 * 18 * 270
           = 1428840

Since GCD(29, 1766107) = 1, by  Euler's Theorem, 38^φ(1766107) =
38^1428840 = 1 (mod 1766107).

29^4286522 = 29^(1428840*3+2) (mod 1766107)
           = (29^1428840)^3 * 29^2 (mod 1766107)
           = 1^3 * 29^2 (mod 1766107)
           = 841 (mod 1766107)
```

5) (10 pts) Given that 2 is a primitive root of 19, determine all primitive roots of 19. Do this problem by hand and show your work and explain your reasoning.

Solution

```
Given that 2 is a primitive root of 19, the set of all primitive roots
of 19 can be represented as 2^k for all k coprime to φ(19).

φ(19) = φ(19^1)
      = 19^1-19^0
      = 18

Values 1 <= k <= 18 coprime to 18:
{1, 5, 7, 11, 13, 17}

2^k = 2^1  = 2  (mod 19)
    = 2^5  = 13 (mod 19)
    = 2^7  = 14 (mod 19)
    = 2^11 = 15 (mod 19)
    = 2^13 = 3  (mod 19)
    = 2^17 = 10 (mod 19)

The primitive roots of 19 are:
```

{2, 3, 10, 13, 14, 15}

6) (20 pts) The notion of a primitive root of a composite number doesn't quite exist in the same way as it does for primes. For example, Euler's Theorem tells us that if gcd(a, 35) =1, then $a^{24} \equiv 1\ (mod\ 35)$. If 35 were to have a "primitive root", then there would be some integer a in between 2 and 33 such that $a^m \not\equiv 1\ (mod\ 35)$, for all integers m, $1 \le m \le 23$. It turns out that no such a exists. Write a short program that proves this assertion and for each integer a, such that gcd(a, 35) = 1 in between 1 and 34, finds the minimum integer m such that $a^m \equiv 1\ (mod\ 35)$, and make a frequency chart of these values of m. **Note: The original question had an error in it. The intention was for the frequency chart to include 24 values not 34 or 35. The only a's that can be tested are a's such that gcd(a, 35) = 1. My guess is students figured this out in the middle of writing the code because the code would infinite loop on any a such that gcd(a, 35) ≠ 1.**

Solution

Cycle	Frequency
1	1
2	3
3	2
4	4
6	6
12	8

Since the code is short, it's just included here:

```
# Arup Guha
# 10/21/2025
# CIS 3362 Hmk #5 Question #6
import math
# To store frequencies.
freq = [0]*25

# Try each base.
for x in range(1, 35):

    if math.gcd(x, 35) != 1:
        continue

    cur = x
    loop = 1

    # Multiply in copies of x till we get to 1.
    while cur != 1:
        cur = (cur*x)%35
        loop+= 1

    freq[loop]+= 1

print("Cycle\tFrequency")
# Print chart.
for i in range(len(freq)):
    if freq[i] == 0:
        continue

    print(i,"\t",freq[i])
```

7) (50 pts) Fermat Theorem Test and Miller-Rabin experimentation

In class we learned that some numbers are good at masquerading as primes with regards to the basic Fermat Theorem Test. Namely, there are composite numbers, n, for which if gcd(a, n) = 1, it's still the case that $a^{n-1} \equiv 1 \ (mod\ n)$ for many choices of a.

For this problem, you'll test a couple ideas we talked about in class experimentally. (I've actually never run these experiments, so I have no idea what the results will be...I know what the theory says, but I actually question it a bit, which is why I've assigned this problem.)

The probability of success of the Miller-Rabin is hinged on the fact that we run it for several potential witness values, a. I am curious as to how many trials are typically needed to discover composite numbers.

Similarly, I want to see how many trials are needed for the Fermat Theorem Test to discover composite numbers.

In theory, both Miller-Rabin and The Fermat Theorem Test might on discover that a number is composite, so we want to keep track of any times, even with 50 repetitions, that either test fails.

Here is the experimental design that I want repeated as many times as possible given the time constraints:

1. Generate a randomly selected composite number NOT divisible by 2, 3 or 5 in between 10^8 and 10^9. (Generate a random odd number in the range and then run the real primality test on it...)

2. Run the Fermat Theorem Test with 50 randomly chosen values of a, tracking how many returned "is probably prime" before receiving the "composite" response. Skip choosing values of a for which gcd(a, n) != 1. (In reality that's proof that n isn't prime, but I want to see the experimental probability that witnesses that actually have a shot report incorrectly.)

3. Run Miller-Rabin with 50 randomly chosen values of a, tracking how many returned "is probably prime" before receiving the "composite" response. Skip choosing values of a for which gcd(a, n) != 1.

Thus, for a single trial, you should record a single integer, in between 0 and 50, inclusive. 0 means that the first value of a tested proved that n was composite. 5 means that the first five values of a test indicated "is probably prime" but the 6^{th} value proved that n was composite. 50 means that the algorithm returned "is probably prime" 50 times in a row so that the overall function erroneously thought that n was prime.

Run as many trials as possible, keeping a frequency chart of how many times, for each algorithm you got an outcome of 0, 1, 2, ..., 50.

Please submit both your code and a nice chart and written summary of your findings. **<u>Please edit my posted Miller-Rabin code.</u>**

Sample Results

Fermat Theorem Test:

# of failures	Frequency
0	999975
1	23
50	2

Miller-Rabin Primality Test:

# of failures	Frequency
0	999998
1	2

The data obtained from this test suggests that the Miller-Rabin primality test is more effective than the Fermat Theorem test. For one, Miller-Rabin was able to deduce that a number was composite in the first trial 99.9998% of the time, with 2 requiring a second trial, but the Fermat Theorem test required a second trial 10 times more often. Furthermore, in 2 cases out of 1,000,000, the Fermat Theorem test was never able to deduce that a composite number was not prime, whereas Miller-Rabin had 100% accuracy. These two tests must have been Carmichael Numbers.

Code will be posted soon.