CIS 3362 Fall 2018 Homework #5 Solutions

1) By using the Python program below, we find $6019208928 = 2^5 \times 3^3 \times 23 \times 101 \times 2999$. It also prints out phi of n, using the second formula for phi of n, where we factor out n.

```
def primefact(n):

    i=2
    phi = n
    while i*i <= n:
        exp = 0
        while n%i == 0:
            n //= i
            exp+=1
        if exp > 0:
            print(i,"^",exp," * ",sep="",end="")
            phi = phi - phi//i
        i += 1

    if n > 1:
        print(n,"^1", sep="")
        phi = phi - phi//n

    print("phi(",n,") = ", phi, sep="")

primefact(6019208928)
```

2) Alternatively, we can work out $\varphi(6019208928)$ by hand:

$$\begin{aligned}\varphi(6019208928) &= \varphi(2^5) \times \varphi(3^3) \times \varphi(23) \times \varphi(101) \times \varphi(2999) \\ &= (2^5 - 2^4)(3^3 - 3^2)(23 - 1)(101 - 1)(2999 - 1) \\ &= 16 \times 18 \times 22 \times 100 \times 2998 = 1899532800\end{aligned}$$

3) 15 and 131 are relatively prime so

$15^{130} \equiv 1 (\text{mod } 131)$, via Fermat's Theorem. It follows that:

$15^{2992} \equiv (15^{130})^{23} \times 15^2 \equiv 1 \times 225 \equiv \mathbf{94\ (mod\ 131)}$

4) 701 and 1224 are relatively prime so by Euler's Theorem we have
$701^{\varphi(1224)} \equiv 1 \ (\text{mod } 1224)$

First, let's prime factorize 1224 by trial division: $1224 = 2^3 \times 3^2 \times 17$.
Thus, $\varphi(1224) = \varphi(2^3) \times \varphi(3^2) \times \varphi(17) = (2^3 - 2^2)(3^2 - 3^1)(17 - 1) = 4 \times 6 \times 16 = 384$.

$701^{2689} \equiv 701^{2688+1} \equiv 701^{2688} \times 701^1 \equiv (701^{384})^7 \times 701 \equiv 1^7 \times 701 \equiv \mathbf{701\ (mod\ 1224)}$

5) $\varphi(17 \times 29) = \varphi(17) \times \varphi(29) = 16 \times 28 = 448$

Thus, we must determine $d = 143^{-1} \pmod{448}$. Use the Extended Euclidean Algorithm:

$$448 = 3 \times 143 + 19$$
$$143 = 7 \times 19 + 10$$
$$19 = 1 \times 10 + 9$$
$$10 = 1 \times 9 + 1$$

$$10 - 9 = 1$$
$$10 - (19 - 1 \times 10) = 1$$
$$2 \times 10 - 1 \times 19 = 1$$
$$2(143 - 7 \times 19) - 1 \times 19 = 1$$
$$2 \times 143 - 14 \times 19 - 1 \times 19 = 1$$
$$2 \times 143 - 15 \times 19 = 1$$
$$2 \times 143 - 15(448 - 3 \times 143) = 1$$
$$2 \times 143 - 15 \times 448 + 45 \times 143 = 1$$
$$47 \times 143 - 15 \times 448 = 1$$

Taking this equation mod 448, we find

$$47 \times 143 \equiv 1 \pmod{448}$$

It follows that **$d = 47$.**

6) We assume that at least one primitive root exists. Let's call this α. We know that of the p-1 values 1, 2, 3, …, p-1, exactly $\varphi(p-1)$ of them share no common factor with p-1, based on the definition of φ.

In order to prove the assertion, we must prove that α^k is a primitive root if and only if gcd(k, p-1) = 1. If we can prove this, then from the list $\alpha, \alpha^2, \alpha^3, \alpha^4, \ldots, \alpha^{p-1}$, the terms that are primitive roots are precisely the terms with the exponents that don't share a common factor with p-1, of which there are exactly $\varphi(p-1)$.

Let gcd(k, p – 1) = 1. We will prove that α^k is a primitive root. We prove this using proof by contradiction. Assume the opposite, that α^k is NOT a primitive root. Then, we must have that two values in the list $\alpha^k, \alpha^{2k}, \alpha^{3k}, \ldots, \alpha^{k(p-1)}$ that are equivalent mod p. Let these two values be α^{ik} and α^{jk}, where $0 < i < j < p$. Thus, we have:

$$\alpha^{jk} \equiv \alpha^{ik} \bmod p$$
$$\alpha^{jk} - \alpha^{ik} \equiv 0 \bmod p$$
$$\alpha^{ik}(\alpha^{jk-ik} - 1) \equiv 0 \bmod p$$

We know that p shares no common factors with α^{ik}.

It follows that p | $\alpha^{jk-ik} - 1$. Thus

$\alpha^{jk-ik} - 1 \equiv 0 \bmod p$
$\alpha^{(j-i)k} \equiv 1 \bmod p$

Since α is a primitive root, we know that the exponent on the left must be a multiple of p – 1:

(p – 1) | (j – i)k.

We know that gcd(p – 1, k) = 1. Thus it follows that (p – 1) | (j – i). But this contradicts the fact that 0 < i < j < p, which means that i >= 1, j <= p-1, so j – i > 0 and j – i ≤ p – 2.

This is our contradiction. It follows that our initial assumption that two values on the given list were equivalent mod p is faulty. If no two of these values are equivalent mod p, we can conclude that α^k is a primitive root.

Now, the second part of the proof is that if gcd(p – 1, k) > 1, then α^k is NOT a primitive root. Let c = gcd(p – 1, k) > 1. Now, consider the term $(\alpha^k)^{(p-1)/c}$ mod p. The exponent $\frac{p-1}{c}$ is clearly less than p – 1. Secondly, this is equivalent to $\alpha^{\frac{k}{c}(p-1)}$ mod p. Notice that c divides evenly into k because c = gcd(k, p – 1), thus c is a divisor of k. Let m = $\frac{k}{c}$, and $m \in Z$. Thus $\alpha^{\frac{k}{c}(p-1)} \equiv (\alpha^{p-1})^{\frac{k}{c}} \equiv 1^m \equiv 1\ (mod\ p)$. This means that α^k isn't a primitive root since raising it to a power less than p – 1 yields 1.

Thus, we have shown that if AND only if gcd(p – 1, k) = 1, is α^k a primitive root of p. Thus, to count the number of primitive roots, we simply look at the list α, α^2, α^3, α^4 , ... , α^{p-1} and count the number of terms that have exponents relatively prime to p – 1. By definition of α, this number is exactly φ(p – 1). As a concrete example, if we know that 2 is a primitive root of p = 19, it follows that 2^1, 2^5, 2^7, 2^{11}, 2^{13}, and 2^{17} are all primitive roots of 19, since 1, 5, 7, 11, 13 and 17 don't share any common factors with 18, p – 1.

Note: This solution is written by Sushant Kulkarni, a past TA of the course.

7) Based on the previous proof, the 16 primitive roots must be 6^{a1}, 6^{a2}, …, 6^{a16}, where each of the values a1, a2, …, a16 are relatively prime with 41 - 1 = 40. Since a1 < a2 < … < a16, it follows that this list is simply the 16 values relatively prime to 40, in between 1 and 40. These are:

1,3,7,9,11,13,17,19,21,23,27,29,31,33,37,39

Thus, we find **a14 = 33, a15 = 37 and a16 = 39.**

Note: $6^3 \equiv 216 \equiv 11$ (mod 41)
$6^{33} \equiv (6^2)^{16}6 \equiv (36)^{16}6 \equiv (-5)^{16}6 \equiv (-125)^5(-5)(6) \equiv (-2)^5(-30) \equiv (-32)(-30) \equiv 960 \equiv$ **17 (mod 41)**
$6^{37} \equiv 6^{33} \times 6^3 \times 6 \equiv 17 \times 11 \times 6 \equiv$ **15 (mod41)**
$6^{39} \equiv 6^{37} \times 6^2 \equiv 15 \times 36 \equiv$ **7 (mod41)**

8) We need to find

Alice Sends to Bob =$13^8 \equiv$ **9 (mod 37),** (using Python)

Bob Sends to Alice = $13^{19} \equiv$ **24 (mod37),** (using Python)

Shared Key = $9^{19} \equiv$ **9 (mod 37)**

Alternatively, the Shared Key = $24^8 \equiv$ **9 (mod 37)**

9) The code that produced the results below is attached as a separate file, FactoringHW5.java.

Number	Pollard-Rho	Trial Division	Fermat
441075437627829133	78 ms	2209 ms	
733561193479131791	16 ms	3974 ms	
611217877192686991	47 ms	3135 ms	
1442059257386438303	47 ms	9069 ms	9437 ms
3008502085141882717	62 ms	13435 ms	906 ms

Since Fermat took much longer on the first three cases, these times are guesstimated below as follows:

A million tests for a were clocked at 8.75 seconds.
Based on the factorizations obtained by both of the other methods, we can determine precisely how many values of a have to be tried before the solution is found and use the scale factor above to guesstimate the time. Here is that chart:

Number	Fermat Guesstimate
441075437627829133	2563 seconds
733561193479131791	1303 seconds
611217877192686991	1610 seconds

Incidentally, the factorizations are provided below:

441075437627829133 = 267583457 * 1648365869

733561193479131791 = 479056439 * 1531262569

611217877192686991 = 398592973 * 1533438667

1442059257386438303 = 1255635859 * 1148469317

3008502085141882717 = 1713733097 * 1755525461

In addition, Safa wrote python code to do all three tests (factor.py) and here are his run-times:

Number	Pollard-Rho1	Pollard-Rho2	Trial Division	Fermat
441075437627829133	601 ms	267 ms	16062 ms	3042906 ms
733561193479131791	479 ms	398 ms	31859 ms	1490593 ms
611217877192686991	384 ms	290 ms	23828 ms	1754468 ms
1442059257386438303	606 ms	380 ms	84390 ms	9953 ms
3008502085141882717	989 ms	475 ms	138203 ms	1000 ms

The Java versions were faster than Python except for Fermat (which probably means my Fermat method is implemented extremely inefficiently!)

In general, for numbers in this range (products of two primes around 10^9), we find that Pollard-Rho is most effective, followed by trial division. Fermat is fairly terrible because the number of steps it takes can come close to trial division, but the individual actions (squaring a, determining if the leftover is a perfect square) take much more time than a single mod (trial division).