# CS 3510: Homework 1C

Due on Saturday, February 5

*Professor Faulkner*

**CS 3510 Staff**

# Problem 1

## Modular Arithmetic (40 points)

### Part A (15 points)

One way of defining congruence modulo $n$ is as follows: given integers $x$, $y$, and $n > 0$, we say that $x \equiv y \mod n$ when $x = y + cn$, with $c$ integer. Prove that if $x \equiv x' \mod n$, then for any polynomial function with integer coefficients $f$, we have $f(x) \equiv f(x') \mod n$. We say $f$ is a polynomial with integer coefficients of degree $k$ if $f$ is of the form $f(x) = \sum_{i=0}^{k} a_i x^i$, where all $a_i$ and $k$ are integers.

### Part B (10 points)

Solve for $x$ in each of the following using modular exponentiation and/or Fermat's little theorem. Show your work. You may NOT use a calculator.

1. $27^{43} \equiv x \mod 127$
2. $2^{2n} \equiv x \mod 3$
3. $5^8 \equiv x \mod 13$

### Part C (10 points)

Use the Extended Euclidean algorithm to find a pair of integers $x$ and $y$ satisfying $57x - 91y = 1$ if such a pair exists.

### Part D (5 points)

Prove or disprove the following statement

$$x^y \equiv x^{y'} \mod n \text{ if } y = y' \mod n$$

### Part A

*Proof.* Given that $f(x) = \sum_{i=0}^{k} a_i x^i = a_k x^k + a_{k-1} x^{k-1} + ... + a_1 x + a_0$, where all $a_i$s and $k$ are integers. Since, $x \equiv x' \mod n$, then $x^p \equiv x'^p \mod n$ where $p$ is an integer. Therefore, $x_p x^p \equiv a_p x'^p \mod n$ where the $a_p$s are integers. Adding these congruences for $p = 0, 1, ..., k$ we have that $A - 0 + a_1 x + a_2 x^2 + ... + a_k x^k \equiv a_0 + a_1' + ... + a_k x'^k \mod n$. Therefore we have that $f(x) \equiv f(x') \mod n$. □

### Part B

1. $27^{43} = (27^2)^{21} * 27 = (729)^{21} * 27 \equiv (94)^{21} * 27 = (94^2)^{10} * 94 * 27 = (8836)^{10} * 94 * 27 \equiv (73)^{10} * 94 * 27 = (73^2)^5 * 94 * 27 = (5329)^5 * 94 * 27 \equiv (122)^5 * 94 * 27 = (122^2)^2 * 122 * 94 * 27 = (14884)^2 * 122 * 94 * 27 \equiv (25)^2 * 122 * 94 * 27 = (625) * 122 * 94 * 27 \equiv 177 * 122 * 94 * 27 = 14274 * 94 * 27 \equiv 50 * 94 * 27 = 4700 * 27 \equiv 1 * 27 \equiv 27 \mod 127$
2. $2^{2n} = (2^2)^n = (4)^n \equiv (1)^n \equiv 1 \mod 3$
3. $5^8 = (5^2)^4 = (25)^4 \equiv (-1)^4 \equiv 1 \mod 13$

### Part C

$$
\begin{aligned}
91 &= 57(1) + 34 \\
57 &= 34(1) + 23 \\
34 &= 23(1) + 11 \\
23 &= 11(2) + 1 \\
1 &= 23 - 11(2) \\
1 &= 23 - (34 - 23(1))(2) \\
1 &= 23(3) - 34(2) \\
1 &= (57 - 34(1))(3) - 34(2) \\
1 &= 57(3) - 34(5) \\
1 &= 57(3) - (91 - 57(1))(5) \\
1 &= 57(8) - 91(5)
\end{aligned}
$$

**Part D**

Consider (1) in Part B. We showed that it is true that $27^{43} \equiv 27 \mod 127$. However, by Fermat's little theorem that $a^p \equiv a \mod p$ and since 127 is prime then $27^{127} \equiv 27 \mod 127$. Additionally then $43 \not\equiv 127 \equiv 0 \mod 127$. Therefore the statement is disproven.

# Problem 2

## RSA Cryptosystem (30 points)

**Part A (20 points)**
Suppose Alice wants to send Bob a message using the RSA scheme.

1. Who should generate the RSA keys?
2. Suppose the person generating the key chooses the primes $p = 13$ and $q = 29$ and the encryption exponent $e = 5$. What must the decryption exponent $d$ be? Show your work. You may use a calculator.
3. If the message being sent is $m = 6$, then what is the encrypted message? Show your work. You may use a calculator.

**Part B (10 points)** Suppose that when you generate your RSA key you pick $p$ and $q$ as random 1024-bit primes, and it turns out that the encryption exponent $e = 5$ works. At first glance, it might seem like using this small value for $e$ is helpful since it would require less computation to encrypt messages. But there's actually a problem here, and you should modify your algorithm to ensure that $e$ is suitably large. Why is this?

(**Hint:** Think about what happens when the message is really short.)

**Part A**

1. The RSA keys should be generated by Bob.
2. So if we take $N = 13 * 29 = 377$, and subsequently, $(13-1)(29-28) = 336$. Then $\gcd(5, 336) = 1$ means we can perform the extended Euclidean algorithm and get that $1 = 336(-4) + 5(269)$. Therefore, the decryption exponent is $d = 269$.
3. To encrypt the message we need to simply do $m^e \mod (N)$ which is $6^5 \mod 377 \equiv 236 \mod 377$. So Alice will send Bob the message 236.

**Part B**
If we use two primes that are 1024-bit then $N$ can be a very large number. If we choose $e$ to be very small like $e = 5$ then we can run the risk that when we process the encryption of $m^e \mod N$ with a relatively small $m$ then it could get confused when trying to decrypt the message. If the message is very small and it is encrypted, then the process is then similarly done to decrypt it by applying $(m^e)^d$ it might not loop back around to the original intended message, it might be large enough $\mod N$.

# Problem 3

## Fermat's Primality Test (30 points)

To answer the following questions, consider the following function. Show your work (you can use a calculator).

```
def naiveIsPrime(N, k):
    for i = 1 to k:
        x = a integer in the range [2, N-1] chosen uniformly at random
        if (x^(n-1) % n) != 1:
            return false
    return true
```

1. (10 points) What is the probability that naiveIsPrime(13, 1) returns true?

   Since 13 is prime by Fermat's Little Theorem it is true that $a^{12} \equiv 1 \mod 13$. Therefore, in this algorithm we know that for any int $x$ in the range $[2, N-1]$ that $x^{13-1} \mod n$ will be $\equiv 1$. So the probability that naiveIsPrime(13,1) returns true is 100%.

2. (10 points) What is the probability that naiveIsPrime(9,1) returns true?

   Since 9 is not prime, we cannot use Fermat's Little Theorem. If we observe the possibilities, we see that $2^8 \equiv 4 \mod 9, 3^8 \equiv 0 \mod 9, 4^8 \equiv 7 \mod 9, 5^8 \equiv 7 \mod 9, 6^8 \equiv 0 \mod 9, 7^8 \equiv 4 \mod 9, 8^8 \equiv 1 \mod 9$. Therefore, the probability that the algorithm returns true is $\frac{1}{8}$

3. (10 points) What is the probability that naiveIsPrime(9,5) returns true?

   We can use the findings from 2 to help with this problem, we simply need naiveIsPrime(9,1) to return true 5 times. Therefore, the probability that algorithm will return true is $\frac{1}{8^5}$.