CS 70 Discrete Mathematics and Probability Theory
Summer 2015 Chung-Wei Lin HW 2

Due Monday July 6 at Noon

1. **Stable Teaching Assistant** (20 points, 5 points for each part)

   You have been asked to assign TAs for the fall semester. Each class has its own method for ranking candidates, and each candidate has their own preferences. An assignment is **unstable** if a class and a candidate prefer each other to their current assignments. Otherwise, it is **stable**.

   Candidate information:

Candidate	CS70 Grade	CS61A Grade	CS61B Grade	Teaching Experience	Overall GPA	Preferences
A	A+	A	A	Yes	3.80	CS70 > CS61A > CS61B
B	A	A	A	No	3.70	CS70 > CS61B > CS61A
C	A	A+	A-	Yes	3.60	CS70 > CS61A > CS61B

   Ranking method:

   - CS70: Rank by CS70 grade. Break ties using teaching experience, then overall GPA.
   - CS61A: Rank by teaching experience. Break ties using CS61A grade, then overall GPA.
   - CS61B: Rank by CS61B grade. Break ties using overall GPA, then teaching experience.

   (a) Find a stable assignment.

   (b) Can you find another, or is there only one stable assignment (if there is only one, why)?

   CS70 is overenrolled and needs two TAs. There is another candidate.

Candidate	CS70 Grade	CS61A Grade	CS61B Grade	Teaching Experience	Overall GPA	Preference
D	A+	A	A+	No	3.90	CS61A > CS61B > CS70

   (c) Find a stable assignment.

   (d) Prove your assignment in Part (c) is stable.

   Answer:

   (a) Use SMA with the following class rankings and students proposing.

CS70	A > C > B
61A	C > A > B
61B	A > B > C

Day	CS70	CS61A	CS61B
1	A, B, C		
2	A	C	B

Assignment: (CS70, A), (CS61A, C), (CS61B, B).

(b) Run SMA with classes proposing and get the same assignment.

Day	A	B	C
1	70,61B		61A
2	70	61B	61A

For any stable assignment, if a student is paired with a class C', the student must prefer his/her optimal class to C' and prefer C' to his/her pessimal class. When students proposed, SMA outputs the student optimal assignment. When classes proposed, SMA outputs the class optimal assignment, which is the student pessimal assignment. Because the student optimal assignment is the same as the student pessimal assignment, C' can only be the class in the assignment for any student, and thus there can only be one stable assignment.

(c) Use SMA with students proposing and rule that CS70 can hold 2 proposals, with the following class rankings:

CS70	A > D > C > B
CS61A	C > A > D > B
CS61B	D > A > B > C

Day	CS70	CS61A	CS61B
1	A, B, C	D	
2	A, C	D	B

Assignment: (CS70, A and C), (CS61A, D), (CS61B, B).

(d) Follow the stability proof in the lecture note to prove that the assignment is stable. Suppose some TA T in the assignment prefers some class C^* to their assigned class C. We will argue that C^* prefers their TA(s) to T, so there cannot be a rogue couple (a class and a TA prefer each other to their current assignments). Since C^* occurs before C in T's list, he must have proposed to it before he proposed to C. Therefore, according to the algorithm, C^* must have rejected him for somebody it prefers. If C^* is not CS70, the Improvement Lemma shows C^* likes its final TA at least as much as T. If C^* is CS70, then we must now prove an alternate Improvement Lemma to show this.

Prove: If T is rejected by CS70 on the k-th day, then every subsequent day $j \geq k$, CS70 has 2 TAs whom it likes at least as much as T.

- *Base case*: On day k, CS70 rejects T, so it must prefer the two TAs it holds.
- *Induction hypothesis*: Suppose claim is true for $j \geq k$
- *Induction step*: On day $j+1$, by induction hypothesis, CS70 has 2 TAs T' and T'' it prefers to T. Either nobody proposes to CS70, or T''' proposes. If T''' is accepted, then it must be preferred over T' or T'', which are both at least as good as T, so T''' is preferred over T.

After proving the alternate Improvement Lemma, we can claim C^* likes its final TA at least as much as T. Therefore, no TA T can be involved in a rogue couple, and thus the assignment is stable.

2. **Stable Roommates Problem** (5 points)

Because people in the US can now marry any gender, we need another way to give them a stable pairing, which is essentially the stable roommates problem. An online dating app Dating70 wants to match up Jerry, Eric, Sam, and Lucy. They give the app the following preferences for their matches:

Jerry	Eric > Lucy > Sam
Eric	Sam > Jerry > Lucy
Sam	Jerry > Eric > Lucy
Lucy	Jerry > Eric > Sam

An algorithm consists of each person x, one by one, proposing to the first person y on their list and executing as follows:

- When y is proposed by x, y crosses off everyone below x on y's list.
- If y holds 2 proposals, y rejects the person y prefers least (crosses off the person on y's list).
- When x is rejected by y, x crosses off y on x's list and proposes to the next person immediately.

This continues until everyone holds exactly one proposal. We start with the following proposals and produce the following table:

Jerry	$\rightarrow$	Eric, Eric crosses off Lucy
Eric	$\rightarrow$	Sam, Sam crosses off Lucy
Sam	$\rightarrow$	Jerry
Lucy	$\rightarrow$	Jerry, Jerry rejects/crosses off Sam, Sam crosses off Jerry
Sam	$\rightarrow$	Eric, Eric rejects/crosses off Jerry, Jerry crosses off Eric
Jerry	$\rightarrow$	Lucy, Lucy crosses off Eric and Sam

Jerry	~~Eric~~ > Lucy > ~~Sam~~
Eric	Sam > ~~Jerry~~ > ~~Lucy~~
Sam	~~Jerry~~ > Eric > ~~Lucy~~
Lucy	Jerry > ~~Eric~~ > ~~Sam~~

Since each person only has a list size of one, the algorithm terminates with the pairing: {(Jerry, Lucy), (Eric, Sam)}.

Now Megan Fox and Angelina Jolie decide to join the group. Try the algorithm on the following table to find a pairing:

Jerry	Eric	>	Megan	>	Lucy	>	Sam	>	Angelina
Eric	Megan	>	Sam	>	Jerry	>	Angelina	>	Lucy
Sam	Jerry	>	Megan	>	Eric	>	Lucy	>	Angelina
Lucy	Angelina	>	Jerry	>	Eric	>	Sam	>	Megan
Megan	Jerry	>	Angelina	>	Eric	>	Lucy	>	Sam
Angelina	Sam	>	Lucy	>	Megan	>	Jerry	>	Eric

Note: The output of this example will be a stable pairing. However, for any instance, if it has a stable pairing, the algorithm cannot guarantee to find the stable pairing. In fact, the algorithm described above is only the Phase 1 of the Irving Algorithm. With the Phase 2, the Irving Algorithm can always find a stable pairing, if the given instance has one. For more information, please check https://en.wikipedia.org/wiki/Stable_roommates_problem

Answer:

Jerry	$\rightarrow$	Eric	Eric crosses off Angelina and Lucy.
Eric	$\rightarrow$	Megan	Megan crosses off Lucy and Sam.
Sam	$\rightarrow$	Jerry	Jerry crosses off Angelina.
Lucy	$\rightarrow$	Angelina	Angelina crosses off Megan, Jerry, and Eric.
Megan	$\rightarrow$	Jerry	Jerry crosses off Lucy and Sam; Jerry rejects Sam.
			Sam crosses off Jerry.
Sam	$\rightarrow$	Megan	Megan rejects Sam.
			Sam crosses off Megan.
Sam	$\rightarrow$	Eric	Eric crosses off Jerry; Eric rejects Jerry.
			Jerry crosses off Eric.
Jerry	$\rightarrow$	Megan	Megan crosses off Angelina and Eric; Megan rejects Eric.
			Eric crosses off Megan.
Eric	$\rightarrow$	Sam	Sam crosses off Lucy and Angelina.
Angelina	$\rightarrow$	Sam	Sam rejects Angelina.
			Angelina crosses off Sam.
Angelina	$\rightarrow$	Lucy	Lucy crosses off Jerry, Eric, Sam, and Megan.

Jerry	~~Eric~~	>	Megan	>	~~Lucy~~	>	~~Sam~~	>	~~Angelina~~
Eric	~~Megan~~	>	Sam	>	~~Jerry~~	>	~~Angelina~~	>	~~Lucy~~
Sam	~~Jerry~~	>	~~Megan~~	>	Eric	>	~~Lucy~~	>	~~Angelina~~
Lucy	Angelina	>	~~Jerry~~	>	~~Eric~~	>	~~Sam~~	>	~~Megan~~
Megan	Jerry	>	~~Angelina~~	>	~~Eric~~	>	~~Lucy~~	>	~~Sam~~
Angelina	~~Sam~~	>	Lucy	>	~~Megan~~	>	~~Jerry~~	>	~~Eric~~

Pairing: (Jerry, Megan), (Eric, Sam), (Lucy, Angelina).

3. **System Equations with Modular Arithmetic** (20 points, 2 points for each part)

   (a) Solve the following system equations:

$$\begin{cases} x+y=9 & \ldots\text{Equation A} \\ x-y=5 & \ldots\text{Equation B} \end{cases}$$

For the following questions (b)–(h), we will consider mod 5, and only numbers in $\{0,1,2,3,4\}$ and notations $\{x,y,+,=\}$ can be used in the answers.

   (b) Rewrite Equation A. (Assume the answer as Equation C.)

   (c) Rewrite Equation B. (Assume the answer as Equation D.)

   (d) Write down $(4\times(\text{Equation C})-(\text{Equation D}))$.

   (e) Solve x from (d).

   (f) Write down $((\text{Equation C})-(\text{Equation D}))$.

   (g) Solve y from (f).

   (h) Rewrite the answer from (a).

   (i) Compare the results from (e), and (h). What do you find?

   (j) Is $(7,7)$ a solution to the original system equations? Is it a solution if we consider mod 5?

Answer:

(a) $(7,2)$. Add A and B to solve for x. Subtract A from B to solve for y.

(b) $x+y=4$, since $9 \equiv 4 \pmod 5$.

(c) $x+4y=0$, since $-1 \equiv 4 \pmod 5$ and $5 \equiv 0 \pmod 5$.

(d) $3x=1$, since $16 \equiv 1 \pmod 5$.

(e) $x=2$, since $2 \times 3 = 6 \equiv 1 \pmod 5$. (Thus, $3^{-1}=2$.)

(f) $2y=4$, since $-3 \equiv 2 \pmod 5$.

(g) $y=2$.

(h) $(2,2)$.

(i) They are the same (mod 5).

(j) No and yes.

4. **Mod Never Bothered Me Anyway** (15 points, 5 points for each part)

(a) Run the Extended Euclid's Algorithm: `egcd`(31,21).

(b) In Arendelle (the name of a fictional country), there are only two types of coins: one worth 31 cents and the other one worth 21 cents. Elsa used coins to buy a gift for Anna and paid 1,010 cents exactly. How many 31-cent coins and 21-cent coins did Elsa use?

(c) Prove: there is only one solution.

Answer:

(a) Running forward:

$$31-21(1)=10;$$
$$21-10(2)=1.$$

Backpropagating:

$$\begin{aligned}1 &= 21-10(2)\\ &= 21-(31-21(1))(2)\\ &= 31(-2)+21(3).\end{aligned}$$

You can also use the following table to get $1=31(-2)+21(3)$:

Function Call	x	y	$\lfloor x/y \rfloor$	$x \bmod y$	Return
#1	31	21	1	10	$(1,-2,3)$
#2	21	10	2	1	$(1,1,-2)$
#3	10	1	10	0	(1,0,1)
#4	1	0	—	—	(1,1,0)

(b) Here, we can use the result from Part (a). Multiplying both sides by 1010, we find:

$$1010 = 31(-2020)+21(3030).$$

This is not a solution because Elsa must use a non-negative number of coins. We can adjust by adding multiples of 21 to the number of 31-cent coins and subtracting multiples of 31 to the number of 21-cent coins.

$$-2020+21(97)=17;$$
$$3030-31(97)=23.$$

Thus, $1010 = 31(17) + 21(23)$.
Note: You can get 97 by $\lceil \frac{2020}{21} \rceil$ or $\lfloor \frac{3030}{31} \rfloor$.

(c) For sake of contradiction, assume there exists another solution $1010 = 31a + 21b$ for some $a, b \in \mathbb{N}$, where $a \neq 17$ or $b \neq 23$. We can subtract this equation from the last equation of Part (b), producing:

$$0 = 31(17-a) + 21(23-b); \tag{1}$$
$$0 \equiv 31(17-a) \pmod{21};$$
$$0 \equiv 21(23-b) \pmod{31}.$$

Part (a) shows $\gcd(31,21) = 1$ (there are no common factors between 31 and 21), so $17 - a \equiv 0 \pmod{21}$ and $23 - b \equiv 0 \pmod{31}$, which implies $a = 17 + 21j$ and $b = 23 + 31k$ for some $j, k \in \mathbb{N}$. Plugging this back into Equation (1):

$$0 = 31(17 - (17 + 21j)) + 21(23 - (23 + 31k));$$
$$j = -k.$$

If $j > 0$, then $k < 0$ and $b = 23 + 31k = (23 - 31j) < 0$, but then $b \notin \mathbb{N}$, which is impossible.
If $j < 0$, then $k > 0$ and $a = 17 + 21j = (17 - 21k) < 0$, but then $a \notin \mathbb{N}$, which is impossible.
If $j = 0$, then $k = 0$, which contradicts our assumption.
Thus, no other non-negative integer solution can be found.

5. **The Bad Bartender** (15 points, 5/10 points for each part)

You're well-known for being a decent bartender at school, so you get hired at an alumni event. The task is deceptively simple — just pour the jungle juice!

The party has started, but you only have a 3-ounce tumbler, a 5-ounce tumbler, and a big bowl of juice. The alumni are very picky — they want an exact amount of liquid in their cups.

You are allowed to pour from one container to another, stopping only when either the pouring container is empty or the receiving container is full. Each of the alumni holds their own cup. You cannot use the alumnus's cup when preparing the exact amount of juice.

(a) Your first alumnus wants 4 ounces of jungle juice. Show how to pour exactly the right amount.

(b) Given an a-ounce tumbler (a is an integer) and a p-ounce tumbler (p is a prime) where $a \neq p \pmod{p}$, design an algorithm to get all possible integer $\{0, 1, \ldots, p\}$ ounces of juice. Reasonably explain why your algorithm works.

Answer:

(a) Fill up the 3-oz tumbler and pour its contents into the 5-oz tumbler. Fill up the 3-oz tumbler again and pour into the 5-oz tumbler until full. Dump out the 5-oz tumbler into the bowl and pour the remaining contents of the 3-oz tumbler (1 ounce) into the 5-oz tumbler. Fill up the 3-oz tumbler and pour its contents into the 5-oz tumbler. The 5-oz tumbler now has 4 ounces of liquid.

(b) Since p is prime and $a \neq p \pmod{p}$, $\gcd(a, p) = 1$. By the Extended Euclid's algorithm, $(r)a + (s)p = 1$ for some integer r, s. Thus, there are integers r_1, s_1 such that $(r_1)a + (s_1)p = n$ for any integer $0 \leq n \leq p$ (we can use $(r_1)a + (s_1)p = p \Longrightarrow (r_1)a + (s_1 - 1)p = 0$ to cover the case $n = 0$). One can always find numbers such that $r_1 > 0$ and $s_1 < 0$. (If either of these conditions is not satisfied, then let $r_1 = r_1 + p$ and $s_1 = s_1 - a$ and repeat until satisfied.) Thus, r_1 will be the total number of a-oz fills and s_1 will be the total number of p-oz dumps.

```
repeat process until done
  while (a-oz tumbler not empty) and (p-oz tumbler not full)
    pour a-oz tumbler's contents into p-oz tumbler
  check if the desired amount is reached
  if (p-oz tumbler is full)
    dump p-oz tumbler into bowl
  else
    fill up a-oz tumbler
```

6. **Generalization of Fermat's Little Theorem** (15 points)

Prove the following generalization of Fermat's Little Theorem: For every positive integer n (not necessarily prime), let S_n be the set of integer $a \in \{1,2,\ldots,n\}$ and $GCD(a,n) = 1$. Then for every $a \in S_n$, we have $a^{|S_n|} \equiv 1 \pmod{n}$. (Here $|S_n|$ denotes the number of elements in S_n.)

Answer: We will try doing the same thing we did in the proof of Fermat's Little Theorem except with the list S_n. We will work in modulo n. Note that we need an extra lemma to show that multiplying things in S_n still gives us things in S_n.

Lemma 1: $\forall s_1, s_2 \in S_n$, $s_1 \cdot s_2 \in S_n$.

Proof. Both s_1 and s_2 share no factors with n because they are relatively prime to it. So the product $s_1 \cdot s_2$ will obviously not share factors with n either, making it relatively prime with n as well. □

Lemma 2: Pick any a in S_n. Then $\forall s_1, s_2 \in S_n$, $a \cdot s_1 \equiv a \cdot s_2 \implies s_1 = s_2$, or in other words, $a \cdot s$ is distinct.

Proof. Suppose for the sake of contradiction that $a \cdot s_1 \equiv a \cdot s_2$ and $s_1 \neq s_n$. Then we have that $a \cdot (s_1 - s_2) \equiv 0$, and because a is relatively prime with n, $a^{-1} \pmod{n}$ exists and we get $s_1 - s_2 \equiv a^{-1} \cdot 0 \equiv 0$, and so $s_1 = s_2$, which is a contradiction, proving Lemma 2. □

Lemma 3: Pick any a in S_n, then $\forall s \in S_n$, $a \cdot s$ is nonzero modulo n. Note that this lemma isn't completely necessary since it is implied by Lemma 1.

Proof. Assume for the sake of contradiction that $a \cdot s \equiv 0$. a is relatively prime with n, so it has an inverse, and so $s \equiv a^{-1} \cdot 0 \equiv 0$, but s can't be 0 because $0 \notin S_n$, and so we've reached a contradiction and proved lemma 3. □

Now onto the main proof.

Proof. Let $S_n = \{s_1, s_2, \ldots, s_k\}$, which is just naming the elements of S_n. Observe that $k = |S_n|$. Pick any $a \in S_n$. Then $R = \{a \cdot s_1, a \cdot s_2, \ldots, a \cdot s_k\}$ are all nonzero, in S_n, and distinct. So the elements of R must be the same as those of S_n. So multiplying all the elements of R and S_n and equating them, we get $a^k s_1 \cdot s_2 \cdot \cdots \cdot s_k \equiv s_1 \cdot s_2 \cdot \cdots \cdot s_k$. Multiplying both sides by $(s_1)^{-1}, \ldots, (s_k)^{-1}$ (which must exist because they are all relatively prime to n), we get $a^k \equiv 1$, or plugging in for k, we have $a^{|S_n|} \equiv 1 \pmod{n}$, proving the generalization. □

7. **RSA** (30 points, 5/5/5/5/10 points for each part)

Consider an RSA scheme modulus $N = pq$, where p and q are prime numbers larger than 3. In this setting, Alice wants to send a message x to Bob with public key (N, e).

(a) Find a condition on p and q such that $e = 3$ is a valid exponent.

(b) Now suppose that $p = 37$, $q = 13$, and $e = 17$. Find the secret key d used in this scheme.

(c) Following Part (b), Bob receives the encrypted message $y = 91$. What was the original message Alice sent (before encrypting it)?

(d) Now suppose that $p = 7$, $q = 3$, and $e = 25$. Alice wants to send a message $x = 8$ to Bob. What is the encrypted message she sends using the public key?

(e) Prove: When $e \equiv 1 \pmod{p-1}$ and $e \equiv 1 \pmod{q-1}$, $x^e \equiv x \pmod{pq}$.

Answer:

(a) We know p and q are primes and that $e = 3$. We also know that $\gcd(e, (p-1)(q-1)) = 1$ as a requirement for RSA, so substituting 3 for e we get $\gcd(3, (p-1)(q-1)) = 1$. This means that 3 does not divide $(p-1)$ and 3 does not divide $(q-1)$. Additionally, since p and q are primes, 3 does not divide p and 3 does not divide q.

For any number $a \in \mathbb{N}$, $a \equiv 0 \pmod 3 \vee a \equiv 1 \pmod 3 \vee a \equiv 2 \pmod 3$. Following this, it must be the case that $p \equiv q \equiv 2 \pmod 3$ (such that $p-1 \equiv q-1 \equiv 1 \pmod 3$). Otherwise, $p-1 \equiv q-1 \equiv 0 \pmod 3$ or $p \equiv q \equiv 0 \pmod 3$, which negates the conclusions that all of p, q, $p-1$, and $q-1$ are not divisible by 3. Since $q \equiv p \equiv 2 \pmod 3$, we can write p and q as different primes of the form $3k+2$, where $k \in \mathbb{N}$ and $k > 0$.

(b) From RSA, $d \equiv e^{-1} \pmod{(p-1)(q-1)}$, so substituting in values for e, p, and q, $d \equiv 17^{-1} \pmod{(36)(12)}$. Then $d \equiv 17^{-1} \pmod{432}$. Use the Extended Euclidean Algorithm to find $17^{-1} \pmod{(432)}$.
Running forward:

$$\text{egcd}(432, 17) = \text{egcd}(17, 7) = \text{egcd}(7, 3) = \text{egcd}(3, 1) = \text{egcd}(1, 0) = 1.$$

Backpropagating:

$$\begin{aligned} 1 &= 1(1) + 0(0) \\ &= 0(3) + 1(1) \\ &= 1(7) - 2(3) \\ &= (-2)(17) + 5(7) \\ &= (5)(432) - 127(17) \end{aligned}$$

So $d \equiv 17^{-1} \equiv -127 \equiv 305 \pmod{432}$.

(c) According to our RSA scheme, for any encrypted message y, $y^d \pmod N$ should reveal the original unencrypted message. Here, $y = 91$, and $N = 13(37) = 481$. so we want to compute $91^{305} \pmod{481}$. Use a calculator!

$$\begin{aligned} 91^{305} &\equiv (91^5)^{61} \equiv 130^{61} \equiv 130(130^{60}) \equiv 130(130^2)^{30} \equiv 130(65)^{30} \\ &\equiv 130((65)^6)^5 \equiv 130(195)^5 \equiv 130(26) \equiv 13 \pmod{481}. \end{aligned}$$

(d) In order to encrypt any message x, compute $x^e \pmod N$ and then is encrypted message. In this case, $x = 8$, $N = pq = 7(3) = 21$, and $e = 25$. Therefore compute $8^{25} \pmod{21}$.

$$8^{25} \equiv 8(8^{24}) \equiv 8(8^4)^6 \equiv 8(1^6) \equiv 8 \pmod{21}.$$

(e) For the proof, we need to make use of the Fermat's Little Theorem and the Chinese Remainder Theorem. By the Fermat's Little Theorem, we know if p is a prime number, $a \in \mathbb{N}$, and $a \not\equiv p \pmod p$, then $a^{p-1} \equiv 1 \pmod p$. According to the Chinese Remainder Theorem, if

$$x = \begin{cases} k_1 & (\text{mod } m_1); \\ k_2 & (\text{mod } m_2); \\ \vdots & \\ k_n & (\text{mod } m_n), \end{cases}$$

where $m_1, m_2, \ldots, m_n$ are pairwise coprime, and $M = m_1 \cdot m_2 \cdot \ldots \cdot m_n$, then we can write:

$$x \equiv y \pmod{n_i}, 1 \le i \le n \iff x \equiv y \pmod M$$

<u>Note</u>: Without the Chinese Remainder Theorem, you can also prove and then use the following claim: given different primes p, q, if $x^{e-1} \equiv k \pmod p$ and $x^{e-1} \equiv k \pmod q$, then $x^{e-1} \equiv k \pmod{pq}$. This can be proved by

$$\begin{aligned}(x^{e-1} \equiv k \pmod p) \wedge (x^{e-1} \equiv k \pmod q) &\implies (p|(x^{e-1} - k)) \wedge (q|(x^{e-1} - k)) \\ &\implies (pq)|(x^{e-1} - k) \\ &\implies x^{e-1} \equiv k \pmod{pq}.\end{aligned}$$

Now onto the main proof.

Proof. Assume that in the RSA scheme $e \equiv 1 \pmod{p-1}$ and $e \equiv 1 \pmod{q-1}$, where p, q are primes. This means we can write $e = 1 + k_1(p-1)$ and $e = 1 + k_2(q-1)$, for some integers k_1, k_2. Therefore,

- For any integer x, $x \not\equiv p \pmod p$, we get $x^e \equiv x^{1+k_1(p-1)} \equiv x\left(x^{(p-1)}\right)^{k_1} \equiv x\left(1^{k_1}\right) \equiv x \pmod p$, by Fermat's Little Theorem.
- For any integer x, $x \equiv 0 \pmod p$, we get $x^e \equiv 0 \equiv x \pmod p$.

Similarly, we can get $x^e \equiv x \pmod q$. Since $x^e \equiv x \pmod p$ and $x^e \equiv x \pmod q$ and p, q are different primes (this is a requirement of RSA), by the Chinese Remainder Theorem, $x^e \equiv x \pmod{pq}$. □