EECS 70 Discrete Mathematics and Probability Theory
Spring 2014 Anant Sahai

Midterm 1

Exam location: 1 Pimentel, back half: SIDs ending in 0 or 5

PRINT your student ID: _____

PRINT AND SIGN your name: _____, _____ _____
(last) (first) (signature)

PRINT your Unix account login: cs70-_____

PRINT your discussion section and GSI (the one you attend): _____

Name of the person to your left: _____

Name of the person to your right: _____

Name of someone in front of you: _____

Name of someone behind you: _____

Section 0: Pre-exam questions (3points)

1. **What other classes are you taking this term? (1 pt)**

2. **What activity do you really enjoy? Describe how it makes you feel. (2pts)**

Do not turn this page until the proctor tells you to do so. You can work on Section 0 above before time starts.

Section 1: Straightforward questions (37 points)

3. Its Own Inverse (5pts)

For $p > 1$, prove that $p-1$ is always its own multiplicative inverse in mod p arithmetic.

Solution: $(p-1)^2 \equiv p^2 - 2p + 1 \equiv p \cdot (p-2) + 1 \equiv 1 \mod p$

EGCD was used by a few students. Using EGCD gives: $p \cdot 1 + (p-1) \cdot (-1) = 1$.

Which tells us that $-1 \equiv p-1 \mod p$ is the inverse of $p-1$

Many students attempted to use induction here, but it was not necessary. Points were not deducted for using induction unless the student also made other errors.

4. Actually Invert it (10pts)

What is the multiplicative inverse of 21 in mod 31 arithmetic?

(You should show that you've gotten the right answer and tell us how you got it, but you can follow whatever path you want to get the answer.)

Solution: The inverse of 21 mod p is 3.

Brute-forcing the answer and then verifying was fairly common:

$$3 \cdot 21 \equiv 63 \equiv 1 \mod 31$$

Another way to find this is to see that $21 \equiv -10 \mod 31$ and $3 \cdot 10 \equiv 30 \equiv -1 \mod 31$, therefore $3 \cdot 21 = 3 \cdot (-10) \equiv -(-1) \equiv 1 \mod 31$.

Another common approach was to explicitly compute the inverse with the EGCD algorithm:

x	y	d	a	b	equation
31	21	1	-2	3	$-2 \cdot (31-21) + 1 \cdot 21 = -2 \cdot 31 + 3 \cdot 21 = 1$
10	21	1	-2	1	$0 \cdot 10 + 1 \cdot (21 - 2 \cdot 10) = -2 \cdot 10 + 1 \cdot 21 = 1$
10	1	1	0	1	$0 \cdot 10 + 1 \cdot 1 = 1$
0	1	1	0	1	$0 \cdot 0 + 1 \cdot 1 = 1$

Thus, $-2 \cdot 31 + 3 \cdot 21 \equiv 3 \cdot 21 \equiv 1 \mod 31$.

And this tells us that 3 is the inverse of 21 mod 31.

5. Gotta match 'em all (12pts)

We are considering matching up Trainers with starter Pokémon. After the great crusade to liberate sentient beings from bondage, no longer can Pokémon be caught and enslaved. Now, it has to be a mutually satisfying voluntary partnership — the Pokémon preferences matter too.

Consider the following set of preferences:

Pokémon	Trainer		
A	2	3	1
B	1	3	2
C	3	2	1

Trainer	Pokémon		
1	A	B	C
2	C	A	B
3	B	C	A

a) **What are the Pokémon-optimal and Trainer-optimal stable pairings?**
(no justification necessary)

Solution: Pokémon-optimal and Trainer-optimal stable pairings can be found by running the propose-and-reject algorithm with each side proposing, respectively. In each case, since each proposer will propose to a different proposee on the first day, the algorithm will terminate on the first day. In each optimal pairing, the proposers will end up with the first choice on their preference list.

Pokémon-optimal: (A, 2), (B, 1), (C, 3)
Trainer-optimal: (A, 1), (C, 2), (B, 3)

b) Insert another Trainer 4 and Pokémon D into the universe and incorporate them into a set of preferences consistent with those given above. *i.e.* The relative ordering between the ones above must still remain the same. (e.g. A must still prefer 2 over 3, etc.) **Do this in some way such that the propose-and-reject algorithm with Trainers proposing and Pokémon rejecting involves all the trainers being rejected at least once.** Run the algorithm to show that this indeed happens for your proposed preferences.

Solution: There are many possible solutions to this problem. Here is a description of how to find a subset of them.

If each trainer needs to be rejected at least once, each trainer needs to be rejected by their first choice. Without considering 4 and D at this point, if 1 proposes to B, then 3 will be rejected. If 3 proposes to C, then 2 will be rejected.

Pokémon	Trainer		
A	2	3	①
B	1	③	2
C	3	②	1

Trainer	Pokémon		
1	A → B		C
2	C → A		B
3	B → C		A

Fortuitously, B is 1's second choice, C is 3's second choice, and A is 2's second choice. There is a nice chain reaction here: if A is 4's first choice, and if A prefers 4 over 1, then 1 will get rejected by A, and propose to B; 3 will get rejected by B, and propose to C; 2 will get rejected by C, and propose to A; if A prefers 2 over 4, then 4 will get rejected by A, and all four Trainers would have been rejected.

Then, there exists a set of possible solutions where:
4's first choice is A
A prefers 2 over 4 over 1
D is not 1, 2, or 3's first or second choice
and other preferences are assigned arbitrarily.

Here is one possible configuration satisfying the above constraints:

Pokémon	Trainer			
A	2	3	4	1
B	1	3	2	4
C	3	2	1	4
D	1	2	3	4

Trainer	Pokémon			
1	A	B	C	D
2	C	A	B	D
3	B	C	A	D
4	A	D	B	C

And the stable marriage algorithm would be run as follows:

Pokémon	Day 1	Day 2	Day 3	Day 4	Day 5
A	1 ④	④	④	② 4	②
B	③	① 3	①	①	①
C	②	②	2 ③	③	③
D					④

6. Prove it by induction (10pts)

The j-th harmonic number is defined as

$$H_j = \sum_{i=1}^{j} \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{j}.$$

So $H_1 = 1, H_2 = 1.5, \ldots$.

Use induction to prove that for any positive integer n,

$$\sum_{j=1}^{n} H_j = H_1 + H_2 + \cdots + H_n = (n+1)H_n - n.$$

Solution:

First we verify the **base case** of $n = 1$:

$$(1+1)H_1 - 1 = 2 \cdot 1 - 1 = 1 = H_1 = \sum_{j=1}^{1} H_j$$

Our **inductive hypothesis** is, for $n = k$:

$$\sum_{j=1}^{k} H_j = (k+1)H_k - k$$

Then we perform the **inductive step** to show that the expression holds for $n = k+1$ as follows:

The crucial step here is to notice that by the definition of the harmonic numbers:

$$H_k = \sum_{i=1}^{k} \frac{1}{i} = \sum_{i=1}^{k+1} \frac{1}{i} - \frac{1}{k+1} = H_{k+1} - \frac{1}{k+1}$$

Now, going back to our expression:

$$\begin{aligned}
\sum_{j=1}^{k+1} H_j &= H_{k+1} + \sum_{j=1}^{k} H_j \\
&= (k+1)H_k - k + H_{k+1} \\
&= (k+1)\left(H_{k+1} - \frac{1}{k+1}\right) - k + H_{k+1} \\
&= (k+1)H_{k+1} - 1 - k + H_{k+1} \\
&= (k+2)H_{k+1} - (k+1)
\end{aligned}$$

Which is what we desire. Thus, the formula holds for all positive integers n.

Section 2: True/False (30 points)

For the questions in this section, determine whether the statement is true or false. If true, prove the statement is true. If false, provide a counterexample demonstrating that it is false.

7. Implications (10pts)

If $P \implies Q$, then $Q \implies P$.

Mark one: TRUE or FALSE.

Solution:

False. Truth table shows a counterexample where P is false and Q is true:

P	Q	P ⇒ Q	Q ⇒ P	(P ⇒ Q) ⇒ (Q ⇒ P)
T	T	T	T	T
T	F	F	T	T
F	T	T	F	F
F	F	T	T	T

The third line in the truth table shows the counterexample. Without including the "(P ⇒ Q) ⇒ (Q ⇒ P)" column in the truth table, you needed to explicitly indicate that the 3rd line was the counterexample in your truth table to get full credit.

Counterexamples could also be given in story form, with an example provided below:
P: It is starry.
Q: It is nighttime.
If it is starry, then it is nighttime. However, that does not imply that if it is nighttime, then it is starry, because it could be a cloudy night.

However, these generally required some sort of explanation or justification to get full credit, as is also provided above. Without an explanation, the solution generally got partial credit.

8. Factorial and Mod (20pts)

If p is prime, then $(p-1) \equiv (p-1)! \mod p$.

Mark one: TRUE or FALSE.

Solution:

True. Here's how we show it.

First, consider the set $S = \{1, ..., (p-1)\}$. Since every element in S is coprime with p, we know that each one has a unique multiplicative inverse (mod p) that is also in the set S. To prove that they are unique, suppose that $a, b \in \{1, ..., (p-1)\}$ have the same multiplicative inverse, m.

$$\begin{aligned} a \cdot m &\equiv 1 \mod p \\ b \cdot m &\equiv 1 \mod p \\ m \cdot (a-b) &\equiv 0 \mod p \\ m \equiv 0 \mod p \text{ or } (a-b) &\equiv 0 \mod p \end{aligned}$$

Since m is not divisible by p, $(a-b)$ must be divisible by p. Since $|(a-b)| < p-2$, $a-b$ must equal zero. Therefore, if a and b have the same multiplicative inverse, then $a = b$.

Note that since every number in S has a unique multiplicative inverse from S, taking the multiplicative inverse is a bijection. There are two cases: a number has a multiplicative inverse (mod p) that is itself, or a number has an inverse (mod p) that is different from itself.

Let's look at some number x in the set S that has an inverse (mod p) that is itself.

$$\begin{aligned} x^2 &\equiv 1 \mod p \\ x^2 - 1 &\equiv 0 \mod p \\ (x+1)(x-1) &\equiv 0 \mod p \\ x+1 \equiv 0 \mod p \text{ or } x-1 &\equiv 0 \mod p \\ x &\equiv \pm 1 \mod p \end{aligned}$$

This shows that only 1 and $p-1$ are their own inverses. Each of the numbers $2, 3, ..., (p-3), (p-2)$ have a unique inverse which is not themselves.

Now going back to the statement $(p-1) \equiv (p-1)!$ (mod p), there are two cases to show.

- $p = 2$: $(2-1) = 1 \equiv 1!$ (mod p). So, true.
- $p > 2$: We know for every such p, p is an odd number. Then there are an even number of terms in $(p-2)(p-3)\cdots(3)(2)$. Since we have shown each number in this term has a unique inverse (mod p), each term will pair up with its multiplicative inverse and result in $1 \cdot 1 \cdots 1 \cdot 1 \equiv (p-2)(p-3)\cdots(3)(2)$ (mod p). We now have $(p-1)!$ (mod p) $\equiv (p-1) \cdot 1 = (p-1)$. So, again, the statement is true.

For both cases, we have shown the statement is true.

9. Make your own EGCD (20pts)

The extended GCD algorithm discussed in the notes leveraged the fact that $GCD(x,y) = GCD(y, x \bmod y)$.

Make an EGCD algorithm based instead on the fact that if $y \leq x$, then $GCD(x,y) = GCD(y, x-y)$. Prove that your algorithm always terminates when given any natural numbers (x,y) and correctly returns a triple of integers (d,a,b) so that $d = GCD(x,y)$ and $d = ax+by$.

Your algorithm should be recursive and you are not allowed to use the mod operation nor division nor multiplication in the algorithm. (You can use whatever you want in your proof.)

Solution: To create the algorithm, we want to use a similar argument to the one used to justify the EGCD algorithm. (See Note 5 for details on this.) This solution creates it step by step, to show how you might arrive at the answer.

First, begin with the base case, when $y = 0$.

```
algorithm extended-gcd(x,y)
  if y = 0 then return (x, 1, 0)
  # Recursive step goes here
```

For the recursive step, there are two cases. Either $x - y < y$, or $x - y \geq y$. Depending on which is true, the recursive call will differ, because the first argument to GCD is assumed to be bigger than the second one.

```
algorithm extended-gcd(x,y)
  if y = 0 then return (x,1,0)
  else
    if x - y < y
      (d, a, b) = extended-gcd(y, x-y)
      # Compute new d,a,b
    else
      (d, a, b) = extended-gcd(x-y, y)
      # Compute new d,a,b
```

This only leaves figuring out what the new d,a,b are. First, do the case where $x - y < y$. Assuming that the recursive call gives d,a,b such that

$$d = ay + b(x-y)$$

Rearranging this gives $d = bx + (a-b)y$, giving the coefficients we want.

For the case where $x - y \geq y$, again assume the recusrive call gives

$$d = a(x-y) + by$$

which can be rearranged into $d = ax + (b-a)y$. This gives the final algorithm

```
1 algorithm extended-gcd(x,y)
2   if y = 0 then return (x, 1, 0)
3   else
4     if x - y < y
5       (d, a, b) = extended-gcd(y, x-y)
6       return (d, b, a-b)
7     else
8       (d, a, b) = extended-gcd(x-y, y)
9       return (d, a, b-a)
```

By similar reasoning other valid functions are

```
algorithm extended-gcd(x,y)
  if y = 0 then return (x, 1, 0)
  else
    if x < y
      (d, a, b) = extended-gcd(y, x)
      return (d, b, a)
    else
      (d, a, b) = extended-gcd(y, x-y)
      return (d, b, a-b)
```

or

```
algorithm extended-gcd(x,y)
  if y = 0 then return (x, 1, 0)
  else
    if x < y
      (d, a, b) = extended-gcd(x, y-x)
      return (d, a-b, b)
    else
      (d, a, b) = extended-gcd(y, x-y)
      return (d, b, a-b)
```

Proof 1:
To prove that the algorithm terminates and the returned triple is correct, we use strong induction on the sum of the inputs, $z = x + y$. Let $P(z)$ be the proposition that $EGCD(x,y)$, where $x + y = z$, terminates and returns the correct triplet. We denote z' as the sum of inputs to the next recursive call.

Base Case: $z = 0$, then the function terminates and returns $(d,a,b) = (0,1,0)$, which is correct.

Inductive Hypothesis: Assume for all $z \leq k$, $P(z)$ is true.

Inductive Step: For $z = k+1$, we want to prove that $P(k+1)$ is true. We consider a call to $EGCD(x,y)$ such that $x + y = z = k+1$. We call either $EGCD(y, x-y)$ (line 5) or $EGCD(x-y, y)$ (line 8). For both cases, we have $z' = x$, the larger input. By the same arguments used in generating the algorithm, the triplets $(d,a,b-a)$ and $(d,b,a-b)$ are correct. We have two cases for which we prove the function terminates.

- $z' = x = k+1$. This means that $y = 0$, so the function must terminate by returning $(x,1,0)$, which is the correct triplet.
- $z' = x \leq k$. For this case, the function terminates and returns the correct triplet by the inductive hypothesis.

Proof 2:
To prove that the algorithm terminates and the returned triple is correct, use strong induction on y. We use the first `EGCD(x,y)` function given above. Let $P(y)$ be the proposition that $EGCD(x,y)$ terminates and is correct for all $x \geq y$.

Base case: $y = 0$. The algorithm will return $(d,a,b) = (x,1,0)$. This gcd is correct, and $x = 1 \cdot x + 0 \cdot y$.

Inductive hypothesis: Assume $P(y)$ is true for all $0 \leq y \leq k$.

Inductive step: We want to prove $P(k+1)$ true, or that $EGCD(x,k+1)$ terminates and is correct for all $x \geq k+1$. There are two cases for the next recursive call. We denote the smaller input to this call as y' and to the subsequent call y''.

Case 1: $x-(k+1) < (k+1)$ (line 4 of the function). The algorithm runs $(d,a,b) = EGCD(k+1, x-(k+1))$. In this case, $y' = x-(k+1) \leq k$, so by the inductive hypothesis the recursive call is correct. By the same argument used to create the algorithm, the triple $(d,b,a-b)$ is correct.

Case 2: $x-(k+1) \geq (k+1)$ (line 7 of the function). The algorithm runs $(d,a,b) = EGCD(x-(k+1), k+1)$. Now $y' = k+1$, and so we cannot invoke the induction hypothesis. We go onto the subsequent recursive call. The algorithm has two inputs, $((k+1), x-2(k+1))$. Again there are two cases. If again $x-2(k+1) < (k+1)$, then $y'' = x-2(k+1) \leq k$ and by the induction hypothesis, the call is correct. If $x-2(k+1) \geq (k+1)$, then $y'' = (k+1)$ and the next recursive call will deal with inputs $(x-3(k+1),(k+1))$. We can see that if $y << x$, this will continue to go on for forms of $(x-n(k+1),(k+1))$, until $x-n(k+1) < (k+1)$, and we can invoke the induction hypothesis. Again by the same argument used in generating the algorithm, the triple $(d,a,b-a)$ is correct.

Thus, by induction, this EGCD algorithm terminates and gives the correct coefficients.

10. Don't let it (this problem) go (20pts)

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?**

(Think about what you have learned that can be applied and write down what you try. We also need to be able to understand your work so please give us comments to go with your calculations.)

(First HINT: Coins as cups.
Second HINT: Zero plus anything is ...
Third HINT: Feel free to use other problems on this exam to help you.
Fourth HINT: Feel free to ignore these hints if they are not helpful.)

Solution: The problem can be formulated as asking for natural numbers x, y such that

$$1010 = 31x + 21y$$

There are two approaches to this problem. The first is to see that this looks somewhat familiar to EGCD. Note that $\gcd(31,21) = 1$, so we can use EGCD to find values x', y' such that

$$1 = 31x' + 21y'$$

After finding x', y', we can multiply both sides by 1010 to get $1010 = 31(1010x') + 21(1010y')$.

Apply the extended Euclid's algorithm `extended-gcd(31,21)`:

Function Calls	(x', y')	x' div y'	x' mod y'
#1	$(31,21)$	1	10
#2	$(21,10)$	2	1
#3	$(10,1)$	10	0
#4	$(1,0)$	—	—

The returned values of all recursive calls are:

Function Calls	(d,a,b)	Returned Values
#4	—	$(1,1,0)$
#3	$(1,1,0)$	$(1,0,1)$
#2	$(1,0,1)$	$(1,1,-2)$
#1	$(1,1,-2)$	$(1,-2,3)$

Therefore, we get $(x', y') = (-2,3)$. Multiplying by 1010 gives $(x,y) = (-2020, 3030)$. Unfortunately, this uses a negative number of 31-cent coins, which is impossible. To deal with this, note that twenty-one 31-cent coins have the same value as thirty-one 21-cent coins. Increasing x by 21 while decreasing y by 31 will not affect the result of $31x + 21y$. By applying this repeatedly, x can be made positive. A little division will give that you can do this 97 times before you run out of 21-cent coins.

$$x + 97 \cdot 21 = 17$$

$$y - 97 \cdot 31 = 23$$

and indeed $17 \cdot 31 + 23 \cdot 21 = 1010$.

The other approach is somewhat similar. The goal is to find x, y such that.

$$1010 = 31x + 21y$$

Now, if $a = b$, then $a = b \mod n$ for any n. So, let's pick mod 31 and mod 21. Finding 1010 mod both gives the pair of equations

$$18 = 0x + 21y \mod 31 \quad (1)$$
$$2 = 31x + 0y \mod 21 \quad (2)$$

which can be rearranged into

$$y = 18 \cdot 21^{-1} \mod 31 \quad (3)$$
$$x = 2 \cdot 31^{-1} \mod 21 \quad (4)$$

From here, you can run EGCD to find the modular inverses of $21 \mod 31$ and $31 \mod 21$. This was done above, giving

$$y = 18 \cdot 3 = 23 \mod 31 \quad (5)$$
$$x = 2 \cdot -2 = 17 \mod 21 \quad (6)$$

and testing $x = 17, y = 23$ will verify that the solution works.

(A final sidenote: the second approach shows that the solution $x = 17, y = 23$ is unique. Any other positive x and y that satisfy the conditions above will give a total of more than 1010 cents, so there is no other way Elsa could have bought the gift.)

11. (Optional) Asking makes the heart grow fonder (20pts)

Among the trolls, men and women have quantitative preferences for each other. So each man has a personal "liking" score for each woman that expresses how much he likes her and vice-versa, each woman has a "liking" score for each man. A higher score is better. Men and women start with some personalized liking-score lists for each other. Initial liking scores are all distinct positive real numbers.

What makes this land interesting is that the scores can change during the courtship process, but become frozen forever once everyone is married. In this land, they follow the male-propose, female-reject method of matching. (i.e. At every day, each troll man proposes to the troll woman he likes best out of all those who haven't rejected him yet while each woman says "maybe" to the current suitor she likes best and rejects the others proposing to her. When nobody is rejected, all the current "maybes" get married.)

Rogue couples are defined to be any whose **final** liking scores for each other are higher than their **final** scores for their spouses. Stable pairings have no rogue couples.

Suppose that each day a man proposes to a woman, his score for her increases by 5 points. Whenever a woman says "maybe" to a man, her score for him increases by 5 points but only if he had any current competition for her hand. (She gives no bonus points for being the only guy who shows up.) Their response to rejections is even more different. When a woman rejects a man, her score for him drops by a factor of 2 but his score for her increases by 6 points. (He wants more who he can't have.)

State and prove an appropriate "improvement lemma" for the troll women.

(HINT: do some examples that correspond to "shallowland" with tightly clustered initial scores.)

Solution: First note that, from the perspective of troll women, "troll rules" have absolutely no effect for troll men: the fact that their liking score increases by 6 when rejected doesn't matter because they'll never propose again anyway, and the fact that their liking score increases by 5 every "maybe" doesn't matter either, because they would have proposed again anyway (although, if their liking score *decreased* by 5 for every "maybe", that would lead to some interesting consequences). We did not require students to make note of this, but it may help you to understand the problem.

Troll Improvement Lemma: If a woman W receives a proposal from man M on day k, then on every subsequent day, she receives a proposal from a man M* whom she likes at least as much as she ever liked M, and will always like at least as much as M.

Proof:
Suppose not. This definitely holds on day k as she likes M on that day as much as she ever liked M.

Then let $j > k$ be the first day in which this statement is not true. Consider day $j-1$. On day $j-1$, W receives a proposal from some M' whom she likes as much as she ever liked M (that is, if M was rejected, she likes M' as much as she liked M just before he was rejected). She will say "maybe" to either M' or she rejects M' for someone else she likes even more than him. Since she likes M' as much or more than she ever liked M, she also likes this other man she selects as much or more than she ever liked M.

On day j, that man (M' or whoever else she liked better) will propose again. Even on day j, W likes this man as much she ever liked M. This is because M's score could not have improved from day $j-1$ to j, unless he was the one on the string the previous day. In that case, he proposes again on day j. Further, no man's score becomes worse if they return to propose again. Therefore she will receive a proposal from someone (say M*) whom she likes at least as much as she ever liked M, which is a contradiction.

Furthermore, note that W cannot reject M* without rejecting M (and then, if M* $\neq$ M, eventually accepting another proposal from M* before rejecting him). Therefore, if M* is ever rejected (and his rating halved), M's rating will also have been halved. Since at one point W liked M* more than M's pre-reject rating, when both ratings are halved, W will still like M* more than M.

Some students went further and showed that in addition to the improvement lemma holding, the resulting marriages are also stable. You should be able to do that given adequate time.