Kevin Chau

Homework 4

Problem 1

1. a, b are commensurable: $\exists g \in \mathbb{R}, \exists k, k' \in \mathbb{N}: a = kg, \; b = k'g$

Show that Euclid's algorithm terminates for a, b.

$GCD(a,b) = GCD(kg, k'g)$

Since g is already a common divisor of a and b, we know the GCD is at least as large as g, and is some integer multiple of g since $k, k' \in \mathbb{N}$.

$$GCD(kg, k'g) = gn \text{ for some } n \in \mathbb{Z}.$$

$n = GCD(k, k')$

$\Rightarrow GCD(a,b) = g \cdot GCD(k, k')$

Since k and k' are natural numbers, and we proved the GCD algorithm terminates for natural numbers, it will also terminate for commensurable real numbers a & b.

2. By the Law of Cosines:

$$(\overline{EB})^2 = (\overline{EA})^2 + (\overline{AB})^2 - 2(\overline{EA})(\overline{AB})\cos(\angle EAB)$$

$$\overline{EA} = \overline{AB}, \quad \angle EAB = 180 - 360/5 = 108°$$

$$\Rightarrow (\overline{EB})^2 = 2(\overline{EA})^2 - 2(\overline{EA})^2 \cos(108)$$

$$= 2(\overline{EA})^2(1 - \cos(108))$$

$$\overline{EB} = \sqrt{2(\overline{EA})^2(1-\cos(108))}$$

$$= \overline{EA}\sqrt{2(1-\cos(108))}$$

$$\approx \overline{EA}\sqrt{2(1+.31)} = \overline{EA}\sqrt{2.62}$$

$$\Rightarrow \boxed{\overline{EB} > \overline{EA} \quad \text{since } \sqrt{2.62} > 1}$$

Since $\angle EAB$ is the largest angle, $\overline{EB}$ must be the largest size.

3. $\angle EAB = \angle ABC = \angle BCD = \angle CDE = \angle DEA = 108°$ [regular pentagon definition]

$\Rightarrow \angle A'EA = \angle B'BA = (180 - 108)/2 = 36°$ [isosceles triangle]

$\Rightarrow \angle DAE = 36°$ [same argument]

$\Rightarrow \angle EA'A = 180 - \angle A'EA - \angle DAE = 108°$ [sum of angles in triangle $= 180°$]

Since $\overline{EB}$ intersects $\overline{DA}$,

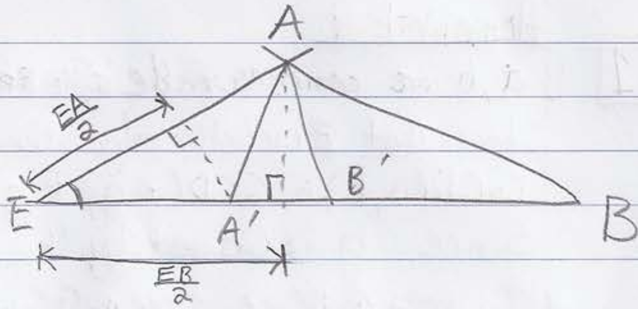
$$\angle EA'A = \angle E'A'B' = 108°$$

Using rotational symmetry, $\angle E'A'B' = \angle A'B'C' = \angle B'C'D' = \angle C'D'E' = \angle D'E'A' = 108°$.

All 5 angles of the inner pentagon are equal (to each other, and $108°$), so $A'B'C'D'E'$ is a regular pentagon by definition.

4. Exploiting similar triangles!

$$\frac{EA'}{E'A'} = \frac{EB}{BC} = \frac{EB}{EA}$$



$$\frac{EA}{\frac{EB}{2}} = \frac{EA'}{\frac{EA}{2}} \Rightarrow EA' = \frac{(EA)^2}{EB}$$

$$E'A' = \frac{EA}{EB} EA' = \boxed{\frac{(EA)^3}{(EB)^2}}$$

$$\Rightarrow \frac{E'B'}{EA'+E'B'} = \frac{EA}{EB} \Rightarrow E'B' = \frac{EA(EA'+E'A')}{EB} = \frac{EA\,EA'}{EB} + \frac{EA\,E'A'}{EB}$$

$$\boxed{E'B' = \frac{(EA)^3}{(EB)^2} + \frac{(EA)^4}{(EB)^3}}$$

5. $$\frac{EB}{EA} = \frac{E'B'}{E'A'} = \left(\frac{(EA)^3}{(EB)^2} + \frac{(EA)^4}{(EB)^3}\right)\left(\frac{(EB)^2}{(EA)^3}\right) = 1 + \frac{EA}{EB}$$

let $EB/EA = x$

$$\Rightarrow \quad x = 1 + \frac{1}{x}$$

$$\Rightarrow x^2 = x+1 \Rightarrow x^2 - x - 1 = 0$$

$$\Rightarrow x = \frac{+1 \pm \sqrt{1-4(1)(-1)}}{2} = \frac{1 \pm \sqrt{5}}{2}$$

so $\frac{EB}{EA}$ is irrational, since $\sqrt{5}$ is irrational.

Since $\frac{EB}{EA}$ cannot be written $\frac{p}{q}$ for $p, q \in \mathbb{Z}$

it follows that $\frac{EB}{EA}$ cannot be written

as $\frac{p}{q} = \frac{pg}{qg}$ for some $g \in \mathbb{R}$.

Hence EB and EA are incommensurable lengths.

Problem 2

(a) Start by assigning man m_i with women w_i for all $i = 1, \dots, n$

$$\Rightarrow S = \{(m_1, w_1), \dots, (m_n, w_n)\}$$

(b) While no rogue pairs:

(a') Pick rogue pair (m^*, w^*)

(b') Swap partners so that m^* and w^* are together

$$S = \{(m^*, w'), (m', w^*), \dots\} \Rightarrow S = \{(m^*, w^*), (w', w^*), \dots\}$$

This Procedure is <u>not</u> guarateed to work; Joe can pick the rogue couples in each step such that a new rogue couple is always created, even while fixing one. So the procedure does not work no matter how Joe picks the rogue couples. Counterexample:

1	—		
2	A	B	C
3	B	A	C

A	3	2	1
B	2	3	1
C	—		

Step 1: A-1 B-2 C-3

(A,2) is a rogue couple

Step 2: A-2 B-1 C-3

(A, 3) is a rogue couple

$\Rightarrow$ A-3 B-1 C-2

(B, 3) is a rogue couple

$\Rightarrow$ A-1 B-3 C-2

(B, 2) is a rogue couple

$\Rightarrow$ $\boxed{\text{A-1 B-2 C-3}}$

So Joe goes back to step 1. If he wants, he can choose to do this loop forever, and there would be no final pairing.

Problem 3

Men Propose to top 2 women.

1.

Men			
1	~~A~~	B	~~C~~
2	A	C	B
3	B	~~C~~	~~A~~

Women			
A	2	3	1
B	3	2	1
C	2	3	1

② = maybe
~~α~~ = crossed off

On the first day, man 1 would propose to A & B, but B will choose 3 over 1 and A will choose 2 over 1.

A	B	C
2 ~~1~~	~~1~~ 3	2 ~~3~~

On the 2nd day, 1 must ask C, who will choose 2 over 1. 1 has no more women to propose to.

A	B	C	
2 ~~3~~	3	2 ~~1~~	⇒ (2-A), (3-B)

But 1 and C are unpaired.

2. For every man that has 2 women who have said "maybe", there is another man who won't be able to pair up with anyone in the end. So when the women dumps the man with another woman, she can pair up with the unpaired men. For n men & women, in the case where n is even, at least n/2 men will pair with n women after the men propose. This leaves n/2 woman to dump the men with 2 proposals, and n/2 men who are unpaired. Then with the traditional algorithm with women proposes, all people will end up paired.

3. Men who propose to 2 women will always end up in a stable pairing, since they are following the regular algorithm. Then the women who propose also create stable pairings because they too are using the traditional algorithm.

4. The resolting pairing is stable because the men will only reject ~~they~~ their current partner if the new proposing woman is better. Now the algorithm really just looks like a female proposing traditional algorithm, which is female optimal.

Problem 4

1. Suppose f is injective ($F(s) = F(s') \Rightarrow s = s'$).
Because F is injective, every unique input s has exactly 1 unique output. This means that after F is applied to a secret, its codeword is unique and identifies its secret exactly, provided one knows the inverse of the injective function. For example, if $f(s)$ adds the string "MySecretIs" then anyone can easily come up with the inverse of f, which is to subtract "MySecretIs".

$f(Adam) =$ MySecretIsAdam

$F(Eve) =$ MySecretIsEve

This is true for any injective (one-to-one function)

2. The conversion formula essentially provides a way to convert a base 26 number, where each digit takes a value from 0-25, corresponding to the letter # in the alphabet, rather than just 0-25. This mapping is injective, so given an $\bar{s}$ and n, we can easily identify the 26^0 digit, 26^1 digit, 26^2 digit, and so on for each letter corresponding to each digit place by following the following algorithm:

   1. start with $\bar{s}$. Floor divide by 26^{n-1}

   $$\bar{\ell}_{n-1} = \left\lfloor \frac{\bar{s}}{26^{n-1}} \right\rfloor$$

   2. To find the next digit, subtract $(\bar{\ell}_{n-1})26^{n-1}$ from $\bar{s}$ (take modulo) and floor divide by 26^{n-2} $\Rightarrow \ell_{n-2} = \left\lfloor \frac{\bar{s} \bmod (26^{n-1})}{26^{n-2}} \right\rfloor = \left\lfloor \frac{\bar{s} - \left\lfloor \frac{\bar{s}}{26^{n-1}} \right\rfloor 26^{n-1}}{26^{n-2}} \right\rfloor$

   3. Keep going by iteratively repeating the above steps until you get to the 26^0 digit. Then

Since $\bar{s}$ is an injective mapping, the above algorithm defines an inverse function to retrieve s.

Proof by strong Induction:

- Base case: $n=1 \Rightarrow \bar{s} = \sum_{k=0}^{0} l_k 26^k = 26^0 \bar{l}_0 \Rightarrow s = l_0$
- Inductive hypothesis:

Assume for any $k \leq n$ that given a unique $\bar{s} = \sum_{k=0}^{n-1} \bar{l}_k 26^k$ and word length n, that s can easily be recovered

- Inductive Step: given $n+1$

$$\bar{s} = \sum_{k=0}^{n+1-1} \bar{l}_k 26^k = \sum_{k=0}^{n} \bar{l}_k 26^k = \bar{l}_n 26^n + \underline{\sum_{k=0}^{n-1} \bar{l}_k 26^k}$$

The second term (that is the sum to $n-1$) is covered by the inductive hypothesis. $\bar{l}_n$ can be found by: $\bar{l}_n = \left\lfloor \frac{\bar{s}}{26^n} \right\rfloor$

Hence $\bar{s}$ can easily be inverted back to s.

$\boxed{\bar{s} = 1110009 \quad n=5}$

$$\bar{l}_4 = \left\lfloor \frac{1110009}{26^4} \right\rfloor = 2$$

$$\bar{l}_3 = \left\lfloor \frac{1110009 - 2(26^4)}{26^3} \right\rfloor = \left\lfloor \frac{196057}{26^3} \right\rfloor = 11$$

$$\bar{l}_2 = \left\lfloor \frac{196057 - 11(26^3)}{26^2} \right\rfloor = \left\lfloor \frac{2721}{26^2} \right\rfloor = 4$$

$$\bar{l}_1 = \left\lfloor \frac{2721 - 4(26^2)}{26} \right\rfloor = \left\lfloor \frac{17}{26} \right\rfloor = 0$$

$$\bar{l}_0 = \left\lfloor \frac{17}{1} \right\rfloor = 17$$

$s = 2, 11, 4, 0, 17$

$\Rightarrow \boxed{s = \text{"CLEAR"}}$

case $\boxed{a \bmod 10 = 2}$

$x_1 \bmod 10 = (4 \bmod 10 + 2 + 1) \bmod 10 = 7 \bmod 10 = 7$

$x_2 \bmod 10 = ((x_1 \bmod 10)(x_1 \bmod 10) \bmod 10 + 7 + 1) \bmod 10 = 17 \bmod 10 = 7$

This reduces to part 1, where 7 is the last digit of a. $k = 1$

case $\boxed{a \bmod 10 = 3}$ Already proved in part 1. $k = 1$ $(k > 0)$

case $\boxed{a \bmod 10 = 4}$ $x_1 \bmod 10 = (4 \cdot 4 \bmod 10 + 4 + 1) \bmod 10 = 11 \bmod 10 = 1$

$x_2 \bmod 10 = (1 \cdot 1 \bmod 10 + 1 + 1) \bmod 10 = 3 \Rightarrow k = 2$

case $\boxed{a \bmod 10 = 5}$ $x_1 \bmod 10 = (5 + 5 + 1) \bmod 10 = 1$

$x_2 \bmod 10 = (1 \cdot 1 \bmod 10 + 1 + 1) \bmod 10 = 3 \Rightarrow k = 2$

$\boxed{a \bmod 10 = 6}$ $x_1 \bmod 10 = (6 + 6 + 1) \bmod 10 = 13 \bmod 10 = 3 \Rightarrow k = 1$

$\boxed{a \bmod 10 = 7}$ $x_1 \bmod 10 = (9 \bmod 10 + 7 + 1) \bmod 10 = 7 \Rightarrow k = 1$

$\boxed{a \bmod 10 = 8}$ $x_1 \bmod 10 = (4 + 8 + 1) \bmod 10 = 3 \Rightarrow k = 1$

$\boxed{a \bmod 10 = 9}$ $x_1 \bmod 10 = (1 + 9 + 1) \bmod 10 = 1$

$x_2 \bmod 10 (1 \bmod 10 + 1 + 1) \bmod 10 = 3 \Rightarrow k = 2$

$\boxed{a \bmod 10 = 0}$ $x_1 \bmod 10 = (0 + 0 + 1) \bmod 10 = 1$

$x_2 \bmod 10 = (1 \cdot 1 \bmod 10 + 1 + 1) \bmod 10 = 3 \Rightarrow k = 2$

Since there are no other digits, we have shown that there always exists a $k > 0$ such that $x_n \bmod 10 =$ constant. For no matter what "a", $k = 2$ is the smallest value that guarantees x_n has a constant last digit.

4. Consider $m=3$

$26 \cdot 1 \bmod 3 = 2$

$26 \cdot 2 \bmod 3 = 1$

$26 \cdot 3 \bmod 3 = 0$

since every integer in mod 3 can be obtained, $m=3$ is safe since it maximizes the number of possible words given $F(s)$.

Consider $m=10$

$26 \cdot 0 \bmod 10 = 0$

$26 \cdot 1 \bmod 10 = 6$

$26 \cdot 2 \bmod 10 = 2$

$26 \cdot 3 \bmod 10 = 8$

$26 \cdot 4 \bmod 10 = 4$

$26 \cdot 5 \bmod 10 = 0$

$26 \cdot 6 \bmod 10 = 6$

$26 \cdot 7 \bmod 10 = 2$

$26 \cdot 8 \bmod 10 = 8$

$26 \cdot 9 \bmod 10 = 4$

Only even number modulo 10 exist in the mod10 universe when 26 is the base. So $m=10$ is unsafe, since it does not maximize the number of words given $F(S)$

5. m is safe if and only if it is coprime with 26.
if m is coprime with 26, then $GCD(m, 26) = 1$. Thus 26 has a multiplicative inverse in modulo m, and thus any integer mod m can be reached since every such number is just a multiple of 1.
If m is safe, we can assume that all integers between 0 and $m-1$ are obtainable. For this to be the case, there must be a multiplicative inverse for 26 in modulo m. Hence $GCD(26, m) = 1$. So 26 and m are coprime.
By showing both directions, we haved proved the original statement.

Problem 5

1. last digit 3 $\quad a_0 = 3$

Simple Induction:

$a = 3 + a_1 10 + a_2 10^2 + a_3 10^3 = \sum_{k=0}^{j} a_k 10^k = 3 + \sum_{k=1}^{j} a_k 10^k$

Base Case: $x_0 = a = 3 + \sum_1^j a_k 10^k$ ends in 3 ✓

Inductive Hypothesis: For $n = k$

Assume $(x_n = x_{n-1}^2 + x_{n-1} + 1) \bmod 10 = 3$

$x_k \bmod 10 = (x_{k-1}^2 + x_{k-1} + 1) \bmod 10 = 3$

Inductive Step:

$x_{k+1} = x_k^2 + x_k + 1$

$x_{k+1} \bmod 10 = (x_k^2 + x_k + 1) \bmod 10$

$= (x_k^2 \bmod 10 + x_k \bmod 10 + 1 \bmod 10) \bmod 10$

$= (x_k^2 \bmod 10 + 3 + 1) \bmod 10$

$= (x_k^2 \bmod 10 + 4) \bmod 10$

$= ((x_k \bmod 10)(x_k \bmod 10) \bmod 10 + 4) \bmod 10$

$= (9 \bmod 10 + 4) \bmod 10 = 13 \bmod 10 = 3$ ✓

Last digit is 3, by Induction

last digit 7:

$a = 7 + \cdots \quad x_0 = a$ ✓

assume: $x_k \bmod 10 = (x_{k-1}^2 + x_{k-1} + 1) \bmod 10 = 3$

$x_{k+1} = x_k^2 + x_k + 1$

$x_{k+1} \bmod 10 = (x_k^2 \bmod 10 + x_k \bmod 10 + 1 \bmod 10) \bmod 10$

$= ([x_k \bmod 10][x_k \bmod 10]^{\bmod 10} + x_k \bmod 10 + 1 \bmod 10) \bmod 10$

$= (49 \bmod 10 + 8) \bmod 10$

$= (9 + 8) \bmod 10 = 17 \bmod 10 = 7$ ✓

Last digit is 7, by Induction.

2) Show there exists a $k > 0$ where last digit of x_n for $n \geq k$ is constant. Proof by cases:

Case $\boxed{a \bmod 10 = 1}$:

$x_1 \bmod 10 = (a^2 + a + 1) \bmod 10 = ((a \bmod 10)(a \bmod 10) \bmod 10 + a \bmod 10 + 1) \bmod 10$

$= (1 + 1 + 1) \bmod 10 = 3$

$x_2 \bmod 10 = [(a^2 + a + 1)^2 + a^2 + a + 2] \bmod 10 = [(3 \cdot 3) \bmod 10 + 1 + 1 + 2] \bmod 10$

$= (9 + 1 + 1 + 2) \bmod 10 = 3$

$x_3 \bmod 10 = [x_2^2 + x_2 + 1] \bmod 10 = [9 \bmod 10 + 3 + 1] \bmod 10 = 3$

this case really just reduces to part 1, where a ended in 3, except now $x_n \bmod 10 = 3$ only for $n \geq k = 1$

3. Prove: $t_{even} + 2t_{odd} \bmod 3 = \sum_{k=0}^{n-1} \bar{l}_k 26^k \bmod 3$

Simple induction on word length n:

Base case: $n=1 \Rightarrow t_{even} = \bar{l}_0 \bmod 3 = \bar{l}_0 26^0 \bmod 3$ ✓

Inductive Hypothesis: Assume true for $n=k$

Inductive step: prove for $n=k+1$

- Lemma: if $c|2 \Rightarrow 26^c \bmod 3 = 1$ (Proved by Induction below)

  $\neg(c|2) \Rightarrow 26^c \bmod 3 = 2$

case $c|2$:

Base: $26^0 = 1 = 1 \bmod 3$

Assume true for $c=k$

$\Rightarrow 26^{k+2} = 26^2 26^k = 1 \cdot 1 \bmod 3 = 1 \bmod 3$ ✓

case $\neg(c|2)$:

Base: $26^1 = 26 \bmod 3 = 2$

Assume true for $c=k$

$\Rightarrow 26^{k+2} = 26^k 26^2 = (1 \cdot 2) \bmod 3 = 2 \bmod 3$ ✓

If k is odd:

$\left(\sum_{k=0}^{k-1} \bar{l}_k 26^k\right) \bmod 3 + \bar{l}_k 26^k \bmod 3$

$= t_{even} + 2t_{odd} \bmod 3 + (\bar{l}_k \bmod 3) 2$

$= t_{even} + 2(t_{odd} + \bar{l}_k) \bmod 3 = t_{even} + 2t'_{odd} \bmod 3$ ✓

If k is even:

$\sum_{k=0}^{k-1} \bar{l}_k 26^k \bmod 3 + \bar{l}_k 26^k \bmod 3$

$= t_{even} + 2t_{odd} \bmod 3 + (\bar{l}_k \bmod 3)(1)$

$= (t_{even} + \bar{l}_k) + 2t_{odd} \bmod 3$

$= t'_{even} + 2t_{odd} \bmod 3$ ✓

Thus by induction we have proved they are equivalent

Problem 6

1. $[x_1, x_2, \ldots, z, \ldots, x_n]$ z is smallest number $x_k > 0$

$\Rightarrow [z, x_1 \bmod z, x_2 \bmod z, \ldots, x_n \bmod z] = [z, y_1, y_2, \ldots, y_n]$

- suppose all $x_k \in [x_1, x_n]$ have a common divisor

$\Rightarrow z = a_k d \quad x_1 = a_1 d \quad x_2 = a_2 d \cdots x_n = a_n d$

For arbitrary $a_1, \ldots, a_n \in \mathbb{N}$

$\Rightarrow x_1 \bmod z = a_1 d \bmod a_k d = a_1 d - \lfloor \frac{a_1 d}{a_k d} \rfloor a_k d =$

$= d(a_1 - \lfloor \frac{a_1}{a_k} \rfloor a_k)$

$\Rightarrow x_n \bmod z = a_n d \bmod a_k d = d(a_n - \lfloor \frac{a_n}{a_k} \rfloor a_k)$

Thus every $y_n = x_n \bmod z = d(a_n - \lfloor \frac{a_n}{a_k} \rfloor a_k)$ has the common divisor d. So every divisor of the set $\{x_n\}$ is also a common divizor of $\{y_n\}$

- Suppose all $y_k \in [y_1, y_n]$ have a common divisor:

$z = b_k d' \quad y_1 = b_1 d' \quad y_2 = b_2 d' \cdots y_n = b_n d'$

For natural numbers $b_1, \ldots, b_n$.

$x_n \bmod z = y_n = x_n - \lfloor \frac{x_n}{z} \rfloor z$

$\Rightarrow x_n = y_n + \lfloor \frac{x_n}{z} \rfloor z = (b_n d' + \lfloor \frac{x_n}{z} \rfloor b_k d')$

$= d'(b_n + \lfloor \frac{x_n}{z} \rfloor b_k)$

since $\lfloor \frac{x_n}{z} \rfloor \in \mathbb{N}$ is natural number, x_n also has common divisor d'. So every divisor of $\{y_n\}$ is a divisor of $\{x_n\}$.

Because every common divisor of $\{x_n\}$ divides $\{y_n\}$ and any common divisor of $\{y_n\}$ is a divisor of $\{x_n\}$, it follows that the greatest common divisor of $\{x_n\}$ is also the greatest common divisor of $\{y_n\}$. This is because if $\{y_n\}$ had any greater divisor, $\{x_n\}$ would have it too.

2. Induction on the list length: (assuming there exists a z)

<u>Base Case</u>: $n=1$

the GCD of a single element is just itself.

$GCD(\{x_1\}) = x_1 = nz[0]$

The algorithm is correct for the base case.

<u>Inductive Hypothesis</u>: Assume for all $k \leq n$ that a list of k elements will correctly go through the GCDmany algorithm

<u>Inductive Step</u>: Let the list size be $n+1$. Since the list is not length 1, it will not run the base case if code. Instead it will recursive compute the GCD of the list where every element is $x_n \bmod z$ where z is the smallest input element. We know that any multiple of z in the list mod z is just 0. So if there are multiples, the recursive GCD call is on a list containing 0 and non-zero elements. Of course, the GCD of a list with 0's is just the GCD of that list without 0's, so we can remove these 0's and reduce the input list to the recursive call by at 1 element, hence the list is covered by the hypothesis, and we will correctly find the GCD using part 1. In the worse case where the input list has no multiples of z, every element in the recursive call must be less than or equal to $z-1$. This means that even in the worse case, the algorithm could repeat until the smallest element is 1, in which case the GCD will be 1.

Thus by strong induction on the length of the list we have proven that the algorithm always works correctly.

3. We need to show that after every recursive call, every number is smaller than itself by at least a factor of 2. If $z \leq \frac{x_n}{2}$ for any x element in the input set, then $x_n \bmod z$ is in the range $[0, z-1]$, which is already less than $x_n/2$. If $z \geq \frac{x_n}{2}$ (but still less than x_n, by definition of z), $x_n \bmod z = x_n - z \leq \frac{x_n}{2}$

So in any case of z relative to any element x, the recursive call decreases the bit size by at least a factor of 2. So if every number has bit size m, it will at most $2m$ calls for the recursion to stop. For a list length n, the number of computations for each recursive step is n. So in total, we have order $O(2mn) = O(mn)$ computations. Though n affects computation time, it does not affect the number of recursive calls. Hence the recursive call bound is: $\boxed{\# \text{recursions} \leq 2m}$

4.
- listing n m-bit numbers take nm computations per recursion
- Minimum of n m-bit numbers: $4mn$ computations per recursion
- $n-1$ modulo operations: $(n-1)(3m^2)$ computations per recursion
- other steps: 10 computations per recursion
- computation takes 1ns

$$\text{total computations per recursion} = nm + 4mn + (n-1)(3m^2) + 10$$
$$= 5mn + 3nm^2 - 3m^2 + 10$$
$$\text{total computations} = (5mn + 3nm^2 - 3m^2 + 10)(2m \text{ recursions})$$
$$= 10m^2n + 3nm^3 - 3m^3 + 20m$$

$n = 100$ $m = 64$

$$\boxed{\begin{aligned} \text{computation time} &\leq 8.1954048 \times 10^7 \text{ ns} \\ &\leq .081954048 \text{ seconds} \end{aligned}}$$

5. $GCD(\{X_n\}) = G = \sum_{i=1}^{n} a_i x_i$

```
                 list
                  ↘
GCD many(list x, linear_combination)
  nz = nonzero elements
  if length(nz) = 1
       return (nz[0], 1, 0)
  [idx, m] = min(nz)
  for each k ≠ idx and 0 ≤ k ≤ len(nz)
           nz[k] = nz[k] mod m
  linear_combination[k] = (b, a - ⌊nz[k]/m⌋ b)

     return GCD(nz, linear_combination)
```

Problem 7

Prove the following properties of GCD:

I) The GCD is <u>commutative</u>:

clearly $GCD(x, y) = GCD(y, x)$ since y and x have the same set of common divisors; no matter if we compute the GCD of x and y or y and x, we will always get the same thing. The algorithm implemented in class is, however, <u>not</u> commutative, because it depends on the assumption in Euclid's algorithm that x is the larger number, so the recursion properly reduces the problem size. We can make our algorithm truly commutative by adding the follow case exception, when the first number is smaller:

```
if x<y:
    return GCD(y,x)
```

II) The GCD is <u>associative</u>:

We need to show $GCD(a, gcd(b,c)) = GCD(GCD(a,b), c)$. Note that if a, b have common divisors, then these common divisors also divide the $GCD(a,b)$.

let $e = GCD(a, gcd(b,c))$

$\Rightarrow e \mid a$ and $e \mid GCD(b,c)$

$\Rightarrow e \mid b$ and $e \mid c$

So the GCD e divides all elements a, b, c

let $F = GCD(GCD(a,b), c)$

$\Leftrightarrow F \mid a, F \mid b, F \mid c$ [F divides all]

Since $e \mid a, b \Rightarrow e \mid GCD(a,b)$ and $e \mid c$, it follows that $e \mid GCD(GCD(a,b), c)$

$\Rightarrow e \mid F$

Similarly, $F \mid b, c \Rightarrow F \mid GCD(b,c) \Rightarrow F \mid GCD(GCD(b,c), a)$

So $F|e$. Since $F|e$ and $e|F$, it follows that $|e| = |F|$, and both $e, F \in \mathbb{N}$; Thus $e = F$, and the two sides being equal proves associativity.

III) Use associativity and commutativity to prove that the GCDmany algorithm is well defined - that is, prove that the GCD of many elements is well defined:

Using associativity and commutaticity, we can always represent any arbitrary length list as a large, nested GCD expression. Of course, because of commutativity, it does not matter what order we nest. Because of associativity, we can nest the GCD such that every GCD call has an element and the GCD of the rest of the elements. This algorithm has the basecase where the GCD call is on 2 elements exactly (rather than 1 element and another GCD call), and properly recurses down to it by making each GCD call wait for the GCD call inside of it to return. Thus we can rewrite the GCD many algorithm from problem 6 as:

manyGCD($\{x_1, x_2, \ldots, x_n\}$):
  return GCD(x_1, GCD(x_2, GCD(x_3 . . .)))

provided we construct a for loop to generate the proper nested expression, and GCD is just the normal Euclid's algorithm.