2824 - Number Theory Week

I have neither given nor received unauthorized assistance_____.

1. (10 pts) These next 2 weeks are about preparing you to build RSA yourself.

Look up the RSA algorithm and **write 3 specific questions or observations** you have about how it works. Khan Academy is a good place to start.
https://www.khanacademy.org/computing/computer-science/cryptography/modern-crypt/v/intro-to-rsa-encryption

You NOT expect to learn RSA from this activity - this is to give you an idea of where we are headed.

One page is fine.

2. (30 pts) Do as many of # 1- 4 p. 255, and #3, #17, #25, #27 from section 4.3 as you need to understand the fundamental concepts on your own (not graded). **On this page and the next**, summarize essential definitions and concepts from Sec. 4.1 - 4.3 . Pictures, concept maps, flash cards, or just a list of ideas are all good. We will grade this **on the appearance of a serious attempt and clear and accurate references** to the material. Really, this is for you.

3. (10pts) Read “The Sieve of Eratosthenes” on page 259. Notice it uses THM 2, when it says “note that the composite integers not exceeding 100 must have a prime factor not exceeding 10.”

    A. Consider Theorem 2 page 258 - show inductively that it is true with some examples.
    B. Re-Prove Theorem 2 page 258 in your own words, **using our proof structure for this proof by contradiction.**

4. (10pts) Recall that we can think of modulo in terms of scoops and cups. **The Euclidean Algorithm can also apply to scoops and cups and solves the question:**

   "Given 2 measuring scoops, of a cups and b cups, what is the smallest measure in cups that can be made?"

   For example, **given a 16 cup measure and a 25 cup measure.**

   Fill 25 and remove 16, you are left with 9 cups.
   Pour the 9 cups *into a new measuring scoop and mark it.*

   Fill the 16 cup and pour into the new 9 cup measure. You are left with 7 cups.
   Pour the 7 cups *into a new measuring scoop and mark it.*

   Fill the 9 cup measure and pour into the new 7 cup measure . You are left with 2 cups.
   Pour the 2 cups *into a new measuring scoop and mark it.*

   Fill the 7 cups measure and pour it into the 2 cups measure 3 times, You are left with 1 cup.

   You now have a one cup measure.

Solve below **(on the following page)**, using the Euclidean Algorithm with scoops **or** the standard Euclidean Algorithm and show your work.:

You must bake a cake and need to measure exactly 1 cup.

Can you make the cake given:

a. 2 scoops measuring 17 and 88?

b. 2 scoops measuring 35 and 88?

c. 2 scoops measuring 37 and 53?

5. (10pts) Following the Square and Mod video, create a “Square and Mod Chart” for powers of 5 mod 17 , then solve the mod problems on the next page with this method.

Show some evidence of the process you are using to get the values below.

5^0 mod 17 =

5^1 mod 17 =

5^2 mod 17 =

5^4 mod 17 =

5^8 mod 17 =

5^16 mod 17 =

5^32 mod 17 =

5^64 mod 17 =

5^128 mod 17 =

5^256 mod 17 =

6. (10pts) Use the chart and the Square ‘n Mod method from the video to find:

    - 5^270 mod 17
    - 5^186 mod 17
    - 5^411 mod 17

Show all steps for each of the 3 problems.

RSA PROJECT CHECK POINT

For the project and the part of the MW, you will need to create a Fast Modular Exponentiation Function (FME).

You have 3 algorithms to choose from:

- Book Version
- Sriram's Version
- Your own based algorithm based on Square'n Mod activities above.

Below are notes/guide to understand the book and Sriram's Algorithm. You may find it helpful to work through both to decide if you want to use one of these or create your own. You will use this code in your project, but for now just write it separately in any IDE you like and focus on getting this code to work.

Then, test your FME function with the book problems.

Finally, for the the graded portion, 9/10, use your code to explore how FME is much faster than regular mod (%).

FME BOOK OPTION

7. (UNGRADED) Annotate Algorithm 5 and example 12 on page 254.
Use the notes below to guide the annotation (either answer on the page or include in your annotation), then answer the questions that follow.

Open with

The algorithm successively finds $b \bmod m$, $b^2 \bmod m$, $b^4 \bmod m, \ldots, b^{2^{k-1}} \bmod m$ and multiplies together those terms $b^{2^j} \bmod m$ where $a_j = 1$, finding the remainder of the product when divided by m after each multiplication. Pseudocode for this algorithm is shown in Algorithm 5. Note that in Algorithm 5 we can use the most efficient algorithm available to compute values of the **mod** function, not necessarily Algorithm 4.

Be sure to reduce modulo m after each multiplication!

ALGORITHM 5 Modular Exponentiation.

procedure *modular exponentiation*(b: integer, $n = (a_{k-1}a_{k-2}\ldots a_1a_0)_2$, m: positive integers)
$x := 1$
$power := b \bmod m$
for $i := 0$ **to** $k - 1$
  if $a_i = 1$ **then** $x := (x \cdot power) \bmod m$
  $power := (power \cdot power) \bmod m$
return x {x equals $b^n \bmod m$}

We illustrate how Algorithm 5 works in Example 12.

EXAMPLE 12 Use Algorithm 5 to find $3^{644} \bmod 645$.

Solution: Algorithm 5 initially sets $x = 1$ and $power = 3 \bmod 645 = 3$. In the computation of $3^{644} \bmod 645$, this algorithm determines $3^{2^j} \bmod 645$ for $j = 1, 2, \ldots, 9$ by successively squaring and reducing modulo 645. If $a_j = 1$ (where a_j is the bit in the jth position in the binary expansion of 644, which is $(1010000100)_2$), it multiplies the current value of x by $3^{2^j} \bmod 645$ and reduces the result modulo 645. Here are the steps used:

$i = 0$: Because $a_0 = 0$, we have $x = 1$ and $power = 3^2 \bmod 645 = 9 \bmod 645 = 9$;
$i = 1$: Because $a_1 = 0$, we have $x = 1$ and $power = 9^2 \bmod 645 = 81 \bmod 645 = 81$;
$i = 2$: Because $a_2 = 1$, we have $x = 1 \cdot 81 \bmod 645 = 81$ and $power = 81^2 \bmod 645 = 6561 \bmod 645 = 111$;
$i = 3$: Because $a_3 = 0$, we have $x = 81$ and $power = 111^2 \bmod 645 = 12{,}321 \bmod 645 = 66$;
$i = 4$: Because $a_4 = 0$, we have $x = 81$ and $power = 66^2 \bmod 645 = 4356 \bmod 645 = 486$;
$i = 5$: Because $a_5 = 0$, we have $x = 81$ and $power = 486^2 \bmod 645 = 236{,}196 \bmod 645 = 126$;
$i = 6$: Because $a_6 = 0$, we have $x = 81$ and $power = 126^2 \bmod 645 = 15{,}876 \bmod 645 = 396$;
$i = 7$: Because $a_7 = 1$, we find that $x = (81 \cdot 396) \bmod 645 = 471$ and $power = 396^2 \bmod 645 = 156{,}816 \bmod 645 = 81$;
$i = 8$: Because $a_8 = 0$, we have $x = 471$ and $power = 81^2 \bmod 645 = 6561 \bmod 645 = 111$;
$i = 9$: Because $a_9 = 1$, we find that $x = (471 \cdot 111) \bmod 645 = 36$.

This shows that following the steps of Algorithm 5 produces the result $3^{644} \bmod 645 = 36$.

Algorithm 5 is quite efficient; it uses $O((\log m)^2 \log n)$ bit operations to find $b^n \bmod m$ (see Exercise 58).

In example 12 on page 254:

a. What is j?

b. What is i?

c. How is the algorithm here similar to Square and Mod?

d. How is it different?

Look at Algorithm 5.

a. What are the inputs?

b. What is x?

c. If $a_i = 1$ what happens in the loop?

d. If $a_i = 0$ what happens in the loop?

FME SRIRAM VIDEO OPTION

8. (UNGRADED) Review Sriram's video where he provides pseudocode for a FME algorithm.

- How is this algorithm is different from the book's algorithm?

- Will the loop in the algorithm terminate? What needs to be added to ensure it will terminate?

9. (10 pts) Choose an FME algorithm to code:
    - Book Version
    - Sriram's Version
    - Your own based on Square 'n Mod

    (You can use section 4.2 exercises p. 255, #25-#28 to test your code.)

Briefly, explain why you choose which algorithm you did.

Include a screenshot of your COMMENTED code (no black backgrounds)

More room for #9 if needed

10. (10pts) To motivate our use of Fast Modular Exponentiation FME, let's do a comparison of two ways we can calculate a^n mod m.

    Write a new function called **NOT_FME** that returns the result of a^n mod m, using just the standard exponent calculation and the built in Python Mod function %.

    Use the variables, a, n, m a^n mod(m).

    Then use the Python function **pow(),** which calculates the same result using FME.

Do your own informal investigation to see if FME has a greater impact on the runtime when n is getting large or when a is getting large. (Hold one variable fixed and then explore with the other, then reverse).

Summarize your results on this page and use as many pages as you need to explore.