MATH 470 Homework 7 (Due October 18 **on Gradescope**)

Instructions:

- FOR 5 POINT QUESTIONS, IF YOU LEAVE AN ANSWER BLANK, YOU WILL AUTOMATICALLY RECEIVE 2/5 POINTS. ANY ATTEMPTED SOLUTION WILL BE GRADED AS USUAL (AND POSSIBLY GET LESS THAN 2/5 POINTS). DOES NOT APPLY TO BONUS QUESTIONS.
- The writing of your homework submission should be done entirely on your own and you should be able to justify all of your writing in your own words.
- Show your work and write legibly. If there is any difficulty in reading or understanding your submission, or if any nontrivial steps are missing in your work, then points may be deducted.
- When you upload your submission to Gradescope, **make sure you match the correct page(s) for each question**. Otherwise your submission may not be graded, or points may be deducted.
- You may use an online calculator or code for the following operations:
  - add, subtract, multiply, quotient, remainder, rounding, non-modular square roots
  - (Extended) Euclidean Algorithm, fast powering

Required problems (in the SECOND edition of textbook: "An Introduction to Mathematical Cryptography", Second Edition, by Hoffstein, Pipher, and Silverman.)

1. (10 points) 3.34(a)
2. (5 points) 3.37(b)
3. (5 points) 3.39(a)
4. (5 points) Recall that $p = 9907$ is a prime. Use quadratic reciprocity to compute $\left(\frac{1002}{9907}\right)$.
5. (5 points) Let p be an odd prime. Prove that the number of quadratic residues modulo p is exactly $\frac{p+1}{2}$.
6. (Bonus 30 points) You may work in groups, but the points will be split evenly amongst the group members. I will accept submissions until Oct 26 (for Q6 only). Email me a copy of your code.

   My RSA public key is (N, e), where

   $$N = 342647387528779375670375098162296213741958911642475645613557064143782 7$$
   $$e = 65537$$

   I receive the following ciphertext:

   $$c = 24005561322298184893056495153466548482984834773346196665912802841267 69$$

   Implement the quadratic sieve to factor N and decrypt the message. You may use a linear algebra library to solve systems of linear congruences, but you should implement everything else from scratch (besides basic arithmetic operations, gcd, and powering).