All exercise numbers from the textbook refer to the **second edition**.

1. Evaluate the discrete logarithm $\log_{40}[33]_{73}$ using the Pohlig-Hellman algorithm, according to the following steps (see the statement of Theorem 2.31 in the textbook for details on the notation). You may use, without proof, the fact that 40 is a primitive root modulo 73.
   (a) Let $N = \text{ord}_{73}[40]_{73}$. Factor N into prime powers as $N = q_1^{e_1} \cdots q_t^{e_t}$.
   (b) Determine the numbers g_i and h_i for each i from 1 to t inclusive. For each i, what is the order of g_i modulo 73?
   (c) For each i, evaluate the discrete logarithm $y_i = \log_{g_i}[h_i]_{73}$, using a method of your choice.
   (d) Solve the system of congruences $x \equiv y_i \pmod{q_i^{e_i}}$ to obtain the discrete logarithm $x = \log_{40}[33]_{73}$.

2. The following sequence of problems will prove part (b) of the theorem from class on Friday 10/14, showing the correctness of the Pohlig-Hellman algorithm in a slightly more general form than Theorem 2.31 in the textbook (in class, I stated the theorem with the hypothesis that the order of g divides N, while the book has the hypothesis that the order of g is exactly equal to N).
   (a) Textbook exercise 1.13.
   (b) Prove that if $r_1, r_2, \cdots, r_t$ are pairwise relatively prime (that is, $\gcd(r_i, r_j) = 1$ for all $i \neq j$), and $N = r_1 r_2 \cdots r_t$, then the t numbers $N/r_1, N/r_2, \cdots N/r_t$ have greatest common divisor 1. (Note: you are proving that the greatest common divisor *of the entire list* is 1; this is not the same as saying that the numbers are *pairwise* relatively prime).
   (c) Suppose that g, h are elements of a group G, N is a positive integer, and that N factors into prime powers as $N = q_1^{e_1} q_2^{e_2} \cdots q_t^{e_t}$. Suppose also that x is an integer such that $g^{x \cdot N/q_i^{e_i}} = h^{N/q_i^{e_i}}$ for $i = 1, 2, \cdots, t$. Deduce from parts (a) and (b) that in fact $g^x = h$.

   Note. The proof of Theorem 2.31 in the textbook will likely be helpful. Note however that you should prove part (c) *without assuming that the order of g is equal to N*, so the argument in the textbook cannot be applied verbatim.

3. Textbook exercise 3.1, parts (a),(b),(c).

4. Textbook exercise 3.5, parts (a),(b),(c),(d).
   Hint for part (c): Prove that $[n]_{MN}$ is a unit if and only if both $[n]_M$ and $[n]_N$ are units. Then apply the Chinese Remainder Theorem.

5. Textbook exercise 3.7.

6. Textbook exercise 3.11, part (a). You should also solve part (b), but you don't need to write it up; instead you will write a program to break the cryptosystem in one of the programming problems.

7. Textbook exercise 3.13.

   Note. You should compare your solution to this problem to your solution to problem 4 on PSet 2. Both use a very similar idea that is quite versatile.

   Programming problems

   Full formulation and submission: `https://www.hackerrank.com/m158-2016-pset-5`

8. Write a program to break the cryptosystem described in problem 3.11. Your program will receive a public key (but not the corresponding private key) and a cipher text, and it should print the original plaintext.

9. Write a program which takes an integer N (possibly up to 1024 bits long) that is guaranteed to factor into prime powers of at most 16 bits in length, and prints those prime power factors. If these prime power factors are $q_i^{e_i}$ (for $i = 1, 2, \cdots, t$), then your program should print them sorted by the value q_i (i.e. in order of the primes, not in order of prime powers).

10. Write a program which solves the a discrete logarithm problem, where the base of the exponentiation has a *known* order considerably smaller than the prime number p. Specifically, your program will read four integers p, g, a, N, where p is a 1024 bit prime, g, a are elements of $\mathbf{Z}/p$, and N is a 32-bit integer guaranteed to be equal to the order of g modulo p. It is further guaranteed that some power of g is congruent to $a \pmod{p}$. Your program should print an element e of $\mathbf{Z}/N$ such that $g^e \equiv a \pmod{p}$.

11. Implement the Pohlig-Hellman algorithm. You have written all of the main ingredients in previous programming problems (including the previous two problems on this assignment). Specifically, you will be given a discrete logarithm problem for with the modulus p is a "weak prime" in the sense that $p - 1$ factors into small prime powers (all 16 bits or smaller).