All exercise numbers from the textbook refer to the **second edition**. Note that some of these problems use terms or ideas that we won't discuss until Monday or Wednesday.

Written problems

1. Write some code to answer the following question: given an integer n, what is the probability that two numbers a, b chosen randomly from 1 to n inclusive satisfy $\gcd(a, b) = 1$? Determine, to six decimal places, this probability for $n = 10,\ 100,\ 1000$, and 10000 (your computer will probably have to think for awhile on the last one). Guess the trend as n goes to infinity.

2. The Euclidean algorithm is built into Python in the `fractions` module. To read its source code and documentation, type the following three commands into the python interpreter.

   ```
   >>> import fractions
   >>> import inspect
   >>> print inspect.getsource(fractions.gcd)
   ```

   State carefully the exact circumstances under which `gcd(a,b)` will return a negative result (the comment in source code gives most of the answer), and explain why. You may need to look up the precise behavior of the operator `%` when negative numbers are used. Briefly explain whether this function will even return 0, and why.

3. Textbook exercise 1.16, parts (a),(b),(c).

4. Textbook exercise 1.19.

5. Textbook exercise 1.24, part (a).

6. Textbook exercise 1.33.

7. Textbook exercise 1.34, parts (c),(d),(e).

8. Textbook exercise 2.4.

9. Textbook exercise 2.5.

Programming problems

Full formulation and submission: `https://www.hackerrank.com/m158-2016-pset-2`

10. Write a problem to solve linear congruences of the form $ax \equiv b \pmod{M}$. If the congruence has solutions, your program should give a single congruence $x \equiv c \pmod{N}$ that describes them all. If the congruence has no solutions, your program must detect this.

11. Write a program that performs Diffie-Hellman key exchange, as in Table 2.2. You will be given p, g, and A, where p is guaranteed to be a 1024-bit prime; you must produce B and the shared secret. You should look up how to generate random numbers (but you do not need to find a "cryptographically secure" way for purposes of this assignment), and the autograder will check to make sure that your program is not deterministic.

12. You play Eve in this problem. You have intercepted six encrypted messages sent from Alice to Bob. The cryptosystem they are using converts a string (the plaintext) into an integer (the ciphertext); so the data you have intercepted consists of six integers. You have also obtained

the source code Alice and Bob are using for their encryption; it is reproduced below on the last page (you can also find it in the Python starter code on hackerrank). Alice and Bob have a secret key k, which is a 1024-bit integer.

Write a program to break Alice and Bob's encryption, and print the original six plaintext messages.

Note. Because Eve accomplishes this attack using only a selection of a few enciphered messages, this is called a *ciphertext-only attack*.

Hint 1. You don't need to focus on what's going on the in `encode` and `decode` functions (they use some syntax that may not be familiar). All you need to know about them is that they convert strings into integers and back. Focus on what happens in the other two functions.

Hint 2. You may find that you can't think of a completely infallible way to break this encryption. That is OK; in this problem (and in the real world) an attack that succeeds with high probability is good enough, and such attacks exist in this case.

```
# Encodes a given string as an integer.
def encode(text):
        code = 0
        for (loc,ch) in enumerate(text):
                code += ord(ch)*(1 << loc*8)
        return code

# Decodes an integer to a string.
def decode(code):
        text = ''
        while code > 0:
                byte = code & 0xFF
                text += chr(byte)
                code >>= 8
        return text

# Enciphers a given string (text) using a secret key (a positive integer).
def encipher(text,key):
        return key*encode(text)

# Deciphers a given integer (cipher) to return the original string.
def decipher(cipher,key):
        return decode(cipher/key)
```