Honors Homework 3

Discrete Structures, CS 173, Spring 2018

Due Monday, April 30th

To do this homework, you'll need to read our handout on RSA and pp. 131-134 from Liebeck, *A Concise Introduction to Pure Mathematics*, 2nd edition, Chapman and Hall, 2006. These are posted on moodle

- A *.rkt file containing all your functions. Include enough comments that I can easily understand what you did.
- A file showing the equation for problem 2, encoded/decode messages and calculations of decoding constants for problems 3 and 5, and the answer to the question in problem 4.
- A file showing sample inputs and outputs for your functions, as well as answers to the encoding/decoding questions. Find inputs and outputs that clearly illustrate that the code is working right.

The non-code files should be a simple standard format such as pdf, docx, plaintext. Screenshots are ok for showing input and output, but combine them all into one file. Similarly, scans of handwritten calculations are ok but must be easy to read and combined into one file. Typing might be easier.

Character encoding

When you convert strings of characters into lists of digits, you should normalize all letters to their uppercase versions, then use this 2-digit encoding of each alphabetic character: A=11, B=12, ..., Z=36. The digits 0-9 should be converted to their ASCII codes, i.e. 0 encodes as 48, 1 encodes as 49, etc. Characters other than letters and digits should be coded as 10.

When converting back from 2-digit codes to letters

- 0 and 10 should convert to space
- codes for letters and digits should convert to the corresponding character
- other numbers (i.e. mistakes) should be translated into #

Problem 1

Write a function that converts an input string to a list of digits, using the encoding described above. It may help to look back at homework 2.

Write the inverse function, which converts a list of digits to a string. Or, not exactly the inverse, because it can't undo the fact that all letters have become uppercase and all miscellaneous characters have been converted to space.

Problem 2

Write an equation expressing b^{2n} in terms of b^n. Write a similar equation expressing b^{2n+1} in terms of b^n.

Using those equations, write a simple recursive function that takes three inputs (b, n, k) and computes $b^n \pmod{k}$. You'll want to have separate cases, depending on whether n is odd or even. To keep intermediate values small, reduce the output mod k at each main step (e.g. each recursive class).

Hint: look at the racket cheat sheet under "arithmetic" for functions like (integer) quotient).

Each time your function executes, it should make only one recursive call (and that call makes one recursive call, and so forth). If your function is calling itself twice, it's much less efficient than it could be.

Problem 3

We're now going to build RSA encoding and decoding functions. For now, let's assume that N has three digits, so each input character is encoded/decoded separately from its neighbors.

Write a Racket function that converts a string of characters to its RSA encoding (i.e. a list of integers). This function should call some of your previously written functions. Also, it should print out the key intermediate step: the list of character codes before they are run through RSA. Use a let form (see the Quick tutorial, section 5) to capture this intermediate result. Then use the functions write and newline to print it.

Also write the inverse function that converts a list of integers back to the corresponding string. Again, print the key intermediate result: the list of character codes.

(a) Encode your netID using the public key $(N, e) = (299, 7)$.

(b) Figure out what d must be, showing key steps in your work. Then decipher the following message to find the name of a data structure used in geometrical algorithms.

$$280, 220, 63, 220, 93, 220, 176, 244, 157, 176, 145, 43, 63, 145, 23$$

Problem 4

When N has four digits, the 2-digit character codes must be regrouped into 3-digit codes. See the example in the Liebeck readings. Write a function that converts a list of 2-digit codes to a list of 3-digit codes.

If the length of the input list isn't divisible by 3, your code should act as if the last value is followed by one or two zero values (as needed). So (34 17 73 58) would be converted to (341 773 580). Notice that this convention is slightly different from the one used by Liebeck. Try to avoid generating extra zeros at the end of the list.

Write a second function that reverses this process, i.e. converts a sequence of 3-digit codes to 2-digit codes. Finally build a second version of your RSA encoding and decoding functions for use with a 4-digit value of N.

Why does our character encoding specify that the digit group 0 should be translated as space rather than flagged as a mistake (#)? What kind of input can produce an output 0? (If you're having trouble figuring this out, do problem 5 first.)

Problem 5

Since James Bond travels first class and doesn't like regular airplane food, so he pre-orders a special desert. Moneypenny has a standing arrangement with British Airways that they can decode these orders using the decoding key $(N, d) = (2911, 221)$. She was on vacation for his latest mission and delegated the job to Q, who confused the decoding and encoding keys. In other words, 221 was really e rather than d.

Figure out the true decoding key d and decrypt the message.

$$2377, 1020, 1652, 1476, 1500, 2000, 141, 1208, 2331$$