CS135 Tutorial 04

Goals

- Apply the design recipe!
- Use the listof-X-template!
- Write lots of list functions!
- Illustrate bottom-up development; talk about top-down development.

Top-Down vs. Bottom-up

	Top-Down	Bottom-up
Advantages	• Pretty sure you'll develop the "right" helper functions. • Might be able to start even if you don't have a clear vision for solving the entire problem.	• Can test as you go.
Disadvantages	• Hard to test until near the end, developing the "bottom" helper functions.	• Might develop helper functions you don't actually need. • Need a clear vision for the entire solution.

Design Recipe

Module 04 Slide 06:

1. Write a draft of the purpose statement
2. Write Examples (by hand, then using `check-expect`)
3. Write Definition Header & Contract
4. Finalize the purpose with parameter names
5. Write Definition Body
6. Write Tests

Caesar Cipher

Given a string, `text`, and a natural number, `shift`, write a function `(encrypt text shift)` that produces a new string encrypted using the Caesar cipher. A Caesar cipher replaces each letter in the `text` with a letter that is `shift` letters away from it in the alphabet.

All characters in `text` must be from the alphabet A-Z (upper case letters) plus space. Space is considered to be the next character after Z.

Note: The Caesar cipher is a well-known encryption method, but it is not secure and can be easily hacked. If you would like to learn more, consider taking CS 458: Computer Security and Privacy.

CQ1: Wrapper Functions

Which of the functions we developed are “wrapper functions”?

```
1. (define (encrypt text shift)
     (list->string (encrypt/lst shift (string->list text))))

2. (define (encrypt/lst n loc)
     (cond [(empty? loc) empty] ...

3. (define (encrypt/char n ch)
     (first (drop n (drop-until ch alpha2))))

4. (define (drop-until ch loc)
     (cond [(empty? loc) empty] ...

5. (define (drop n loc)
     (cond [(= n 0) loc] ...
```

A. All of them
B. None of them
C. 2, 4, 5
D. 1, 2
E. 1, 3