CS135 Tutorial 04

Review of list functions

List Values	List Functions
`empty`: an empty list `(cons v lst)`: where `v` is a value and `lst` is a list value (which includes `empty`)	`(cons v lst)`: Consumes a value and a list; produces a new, longer list. `(first (cons a lst)) => a` `(rest (cons a lst)) => lst` `(empty? empty) => true` `(empty? a) => false` where `a` is any value other than `empty` `(cons? (cons a lst)) => true` `(cons? a) => false` where a is any Racket value not created using `cons`

`a` and `lst` are values. `a` can be any value; `lst` is a list value.

Creating List Values

- Write Racket code for this list:

2	"Hello"	'up

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 CO 487: Applied Cryptography.

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.