Lisp

Commentary

I was able to skip the first steps of reading the getting started thanks to your lesson in class. It was pretty helpful. I expected there to be some way to avoid having to be too explicit with the parens but ultimately I couldn't figure it out, and probably for my own benefit that I put a parens everywhere I possibly could have. My basic approach was to take the letter and move it one step through the pipeline at a time, slowly building out the parentheses chain. Once I had the rot function I looked up how to do a string map and that was pretty easy, encrypt and decrypt done. Solve was definitely the most difficult part. I had to make it recursive because I couldn't figure out how to make it loop without it not being functional. This does the job, I guess.

Google Searches

- char code common lisp
- lisp ubuntu install
- sbcl run file
- common lisp map over string
- common lisp run x times
- common lisp for each
- common lisp print each time through loop
- common lisp recursion

Caesar Implementation

```
;; sbcl --script caesar.lisp
(defun rot (chr shift)
    (if (and (>= (char-code chr) 65) (<= (char-code chr) 90))
        (code-char (+ (mod (+ shift (- (char-code chr) 65)) 26) 65))
        (code-char (char-code chr))
    )
)

(defun encrypt (word shift)
    (map 'string #'(lambda (x) (rot x shift)) (coerce (string-upcase word)
'list))
)

(defun decrypt (word shift)
    (encrypt word (- 26 shift))
)

(defun solve (word maxShift)
    (if (/= maxShift 0)
        (cons (decrypt word maxShift) (solve word (- maxShift 1))
    ))
)
```

```
(print (encrypt "Attack at Once" 4))
(print (decrypt "EXXEGO EX SRGI" 4))
(print (solve "abcdeFGHIJKLmnopqrstuvwxyz ,?;{[()]}" 26))

;2 hours
```

Output

```
"EXXEGO EX SRGI"
"ATTACK AT ONCE"
(
    "ABCDEFGHIJKLMNOPQRSTUVWXYZ ,?;{[()]}"
    "BCDEFGHIJKLMNOPQRSTUVWXYZA ,?;{[()]}"
    "CDEFGHIJKLMNOPQRSTUVWXYZAB ,?;{[()]}"
    "DEFGHIJKLMNOPQRSTUVWXYZABC ,?;{[()]}"
    "EFGHIJKLMNOPQRSTUVWXYZABCD ,?;{[()]}"
    "FGHIJKLMNOPQRSTUVWXYZABCDE ,?;{[()]}"
    "GHIJKLMNOPQRSTUVWXYZABCDEF ,?;{[()]}"
    "HIJKLMNOPQRSTUVWXYZABCDEFG ,?;{[()]}"
    "IJKLMNOPQRSTUVWXYZABCDEFGH ,?;{[()]}"
    "JKLMNOPQRSTUVWXYZABCDEFGHI ,?;{[()]}"
    "KLMNOPQRSTUVWXYZABCDEFGHIJ ,?;{[()]}"
    "LMNOPQRSTUVWXYZABCDEFGHIJK ,?;{[()]}"
    "MNOPQRSTUVWXYZABCDEFGHIJKL ,?;{[()]}"
    "NOPQRSTUVWXYZABCDEFGHIJKLM ,?;{[()]}"
    "OPQRSTUVWXYZABCDEFGHIJKLMN ,?;{[()]}"
    "PQRSTUVWXYZABCDEFGHIJKLMNO ,?;{[()]}"
    "QRSTUVWXYZABCDEFGHIJKLMNOP ,?;{[()]}"
    "RSTUVWXYZABCDEFGHIJKLMNOPQ ,?;{[()]}"
    "STUVWXYZABCDEFGHIJKLMNOPQR ,?;{[()]}"
    "TUVWXYZABCDEFGHIJKLMNOPQRS ,?;{[()]}"
    "UVWXYZABCDEFGHIJKLMNOPQRST ,?;{[()]}"
    "VWXYZABCDEFGHIJKLMNOPQRSTU ,?;{[()]}"
    "WXYZABCDEFGHIJKLMNOPQRSTUV ,?;{[()]}"
    "XYZABCDEFGHIJKLMNOPQRSTUVW ,?;{[()]}"
    "YZABCDEFGHIJKLMNOPQRSTUVWX ,?;{[()]}"
    "ZABCDEFGHIJKLMNOPQRSTUVWXY ,?;{[()]}"
) %
```

Log

Estimate: 2 hours

Date	Hours Spent	Accomplishments
4/6	.25	Install lisp on ubuntu
4/6	.5	Create rot function

Date	Hours Spent	Accomplishments
4/6	.5	Create encrypt/decrypt
4/6	.75	Create solve function
4/6	.5	Debug non-alpha characters

Discrepancy of time

I think I was still pretty close on this one. If it weren't for handling other characters I had the program working in the expected amount of time. I think as a consultant you've got to learn to respect your own time and cut the corners. Maybe I shouldn't have gone through the extra effort of ignoring extra characters, that probably wasn't a real requirement, I did it anyway, oh well.

Overall Review

This language is terrible. It's only purpose should be to demonstrate how functional programming works. It's readability and writability suffer greatly from all the parentheses. I think a language should have formatting standards that are enforced by the language (my favorite python feature), which was made even more apparent by the documentation of this one online. Everyone seems to write their code in different ways, greatly detracting from readability.

Ratings

Readability: 3/10

Writability: 2/10

Ranking: 5/5