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

ML

Commentary

Once again, your lesson in class was a big help for jump starting these programs. After that class I still needed to figure out how to convert to uppercase, test if a character is a letter, and figure out how to map over a string. Although, there is not a great official documentation for the language, alot of the things I was searching for was already on stack overflow, something I cannot say the same for COBOL. I find it interseting that each of these languages use a different word for function, and then have a different second word for an anonymous function. The program was actually pretty simple to write after I had done it a few times. No really big frustrations on this one. Main things that kept me delayed was just finding the new syntax.

Google Searches

- ML install
- ml install ubuntu
- polyml ubuntu run script
- ml get char code
- ml to uppercase
- ml string map
- ml map over string
- ml enumerate
- ml foreach
- ml run x times

Caesar Implementation

```
fun rot (char, shift) = if ((ord (Char.toUpper char)) >= 65 andalso (ord
(Char.toUpper char)) <= 90) then
        (chr (((ord (Char.toUpper char)) - 65 + shift) mod 26 + 65))
    else
        char;

fun encrypt (word, shift) = (String.map (fn ltr => rot(ltr, shift)) word);
fun decrypt (word, shift) = encrypt(word, 26-shift);
fun solve (word, maxShift) = map
    (fn x => print(Int.toString(x) ^ "\t" ^  (decrypt(word, x)) ^ "\n"))
    (List.tabulate(maxShift, fn x => x)); (* run it maxShift times *)

print (encrypt("ATTACK AT ONCE", 4) ^ "\n");
print (decrypt("EXXEGO EX SRGI", 4)^ "\n");
solve("abcdeFGHIJKLmnopqrstuvwxyz ,?;{[()]}", 26);
```

Output

```
EXXEGO EX SRGI
ATTACK AT ONCE
0       ABCDEFGHIJKLMNOPQRSTUVWXYZ ,?;{[()]}
1       ZABCDEFGHIJKLMNOPQRSTUVWXY ,?;{[()]}
2       YZABCDEFGHIJKLMNOPQRSTUVWX ,?;{[()]}
3       XYZABCDEFGHIJKLMNOPQRSTUVW ,?;{[()]}
4       WXYZABCDEFGHIJKLMNOPQRSTUV ,?;{[()]}
5       VWXYZABCDEFGHIJKLMNOPQRSTU ,?;{[()]}
6       UVWXYZABCDEFGHIJKLMNOPQRST ,?;{[()]}
7       TUVWXYZABCDEFGHIJKLMNOPQRS ,?;{[()]}
8       STUVWXYZABCDEFGHIJKLMNOPQR ,?;{[()]}
9       RSTUVWXYZABCDEFGHIJKLMNOPQ ,?;{[()]}
10      QRSTUVWXYZABCDEFGHIJKLMNOP ,?;{[()]}
11      PQRSTUVWXYZABCDEFGHIJKLMNO ,?;{[()]}
12      OPQRSTUVWXYZABCDEFGHIJKLMN ,?;{[()]}
13      NOPQRSTUVWXYZABCDEFGHIJKLM ,?;{[()]}
14      MNOPQRSTUVWXYZABCDEFGHIJKL ,?;{[()]}
15      LMNOPQRSTUVWXYZABCDEFGHIJK ,?;{[()]}
16      KLMNOPQRSTUVWXYZABCDEFGHIJ ,?;{[()]}
17      JKLMNOPQRSTUVWXYZABCDEFGHI ,?;{[()]}
18      IJKLMNOPQRSTUVWXYZABCDEFGH ,?;{[()]}
19      HIJKLMNOPQRSTUVWXYZABCDEFG ,?;{[()]}
20      GHIJKLMNOPQRSTUVWXYZABCDEF ,?;{[()]}
21      FGHIJKLMNOPQRSTUVWXYZABCDE ,?;{[()]}
22      EFGHIJKLMNOPQRSTUVWXYZABCD ,?;{[()]}
23      DEFGHIJKLMNOPQRSTUVWXYZABC ,?;{[()]}
24      CDEFGHIJKLMNOPQRSTUVWXYZAB ,?;{[()]}
25      BCDEFGHIJKLMNOPQRSTUVWXYZA ,?;{[()]}
```

Log

Estimate: 2 hours

Date	Hours Spent	Accomplishments
4/6	.25	Install local version of ML
4/6	.25	Create rot function
4/6	.25	Create encrypt/decrypt
4/6	.5	Create solve
4/20	.5	Fix special characters

Discrepancy of time

I basically finish all of these programs at the same time period. If I ever get ahead it because I either skipped something or I'm just yet to have the inevitable blocker. For example here, I was done with encrypt and decrypt in 45 minutes, but the solve and getting special characters to work was a much longer process.

Overall Review

Even though ML went pretty well for me, I would only rank the language itself over lisp. There also wasn't a lot of documentation and I would have been stuck for much longer without your basic rundown. Erlang is definitely better and JS and Scala are the only legitimate programming languages in the assignment. Most of the operators are what I would expect them to be, aside from andalso. Who in there right mind came up with that? I think I read that or is referred to as orelse???? Things like this are what lead to a costly language. Erlang, I believe is a worse offender but ML definitely has it's quirks.

Ratings

Readability: 6/10

Writability: 6/10

Ranking: 4/5

Erlang

Commentary

I began this language how I start all languages, look up their getting started page. Despite having a getting started page, it is very unhelpful on its own. It almost appears to be a broken link. It's clear the people that made this language were not big on web development, one of the reasons JS documentation is so plentiful online. Another quirky thing about the language is that you can't really run a file the same way you can in LISP and ML, you must first start a shell in the same directory as a .erl file, then compile, then run the functions that you choose to expose in the file. Very strange design, not sure I completely understand why that choice was made either. One thing that was great, it had the easiest install of all the languages as ubuntu's apt package manager had the erlang package officially supported.

Google Searches

- erlang getting started
- erlang ubuntu
- erlang run file
- erlang examples
- erlang get char code
- erlang from char code
- erlang if
- erlang map
- erlang map over x times
- erlang foreach

Caesar Implementation

```
-module(caesar).
-export([encrypt/2, decrypt/2, solve/2]).

rot(Ltr, Shift) when hd(Ltr) >= 65, hd(Ltr) =< 90 ->
    (hd(Ltr) - 65 + Shift) rem 26 + 65;
rot(Ltr, _) ->
    hd(Ltr).

encrypt(Word, Shift) ->
    lists:map(fun(Ltr) -> rot([string:to_upper(Ltr)], Shift) end, Word).

decrypt(Word, Shift) ->
    encrypt(Word, 26 - Shift).

solve(Word, Shift) ->
    lists:foreach(
        fun(X) -> io:format("Shift: ~p\tResult: ~p ~n", [X, decrypt(Word,
X)]) end,
```

```
        lists:seq(1, Shift)
    ).
```

Output

```
3> caesar:encrypt("ATTACK AT ONCE", 4).
"EXXEGO EX SRGI"
4> caesar:decrypt("EXXEGO EX SRGI", 4).
"ATTACK AT ONCE"
5> caesar:solve("abcdeFGHIJKLmnopqrstuvwxyz ,?;{[()]}", 26).
Shift: 1        Result: "ZABCDEFGHIJKLMNOPQRSTUVWXY ,?;{[()]}"
Shift: 2        Result: "YZABCDEFGHIJKLMNOPQRSTUVWX ,?;{[()]}"
Shift: 3        Result: "XYZABCDEFGHIJKLMNOPQRSTUVW ,?;{[()]}"
Shift: 4        Result: "WXYZABCDEFGHIJKLMNOPQRSTUV ,?;{[()]}"
Shift: 5        Result: "VWXYZABCDEFGHIJKLMNOPQRSTU ,?;{[()]}"
Shift: 6        Result: "UVWXYZABCDEFGHIJKLMNOPQRST ,?;{[()]}"
Shift: 7        Result: "TUVWXYZABCDEFGHIJKLMNOPQRS ,?;{[()]}"
Shift: 8        Result: "STUVWXYZABCDEFGHIJKLMNOPQR ,?;{[()]}"
Shift: 9        Result: "RSTUVWXYZABCDEFGHIJKLMNOPQ ,?;{[()]}"
Shift: 10       Result: "QRSTUVWXYZABCDEFGHIJKLMNOP ,?;{[()]}"
Shift: 11       Result: "PQRSTUVWXYZABCDEFGHIJKLMNO ,?;{[()]}"
Shift: 12       Result: "OPQRSTUVWXYZABCDEFGHIJKLMN ,?;{[()]}"
Shift: 13       Result: "NOPQRSTUVWXYZABCDEFGHIJKLM ,?;{[()]}"
Shift: 14       Result: "MNOPQRSTUVWXYZABCDEFGHIJKL ,?;{[()]}"
Shift: 15       Result: "LMNOPQRSTUVWXYZABCDEFGHIJK ,?;{[()]}"
Shift: 16       Result: "KLMNOPQRSTUVWXYZABCDEFGHIJ ,?;{[()]}"
Shift: 17       Result: "JKLMNOPQRSTUVWXYZABCDEFGHI ,?;{[()]}"
Shift: 18       Result: "IJKLMNOPQRSTUVWXYZABCDEFGH ,?;{[()]}"
Shift: 19       Result: "HIJKLMNOPQRSTUVWXYZABCDEFG ,?;{[()]}"
Shift: 20       Result: "GHIJKLMNOPQRSTUVWXYZABCDEF ,?;{[()]}"
Shift: 21       Result: "FGHIJKLMNOPQRSTUVWXYZABCDE ,?;{[()]}"
Shift: 22       Result: "EFGHIJKLMNOPQRSTUVWXYZABCD ,?;{[()]}"
Shift: 23       Result: "DEFGHIJKLMNOPQRSTUVWXYZABC ,?;{[()]}"
Shift: 24       Result: "CDEFGHIJKLMNOPQRSTUVWXYZAB ,?;{[()]}"
Shift: 25       Result: "BCDEFGHIJKLMNOPQRSTUVWXYZA ,?;{[()]}"
Shift: 26       Result: "ABCDEFGHIJKLMNOPQRSTUVWXYZ ,?;{[()]}"
```

Log

Estimate: 2 hours

Date	Hours Spent	Accomplishments
4/6	.1	Set up erlang shell and run hello world
4/6	.5	Create rot function
4/6	.5	Create encrypt/decrypt
4/6	.25	Create solve function

Date	Hours Spent	Accomplishments
4/13	.25	Fix for special characters

Discrepancy of time

I basically take the same amount of time for each of these languages that I haven't used before. Therefore, I've gotten quite good at guessing the time it takes me.

Overall Review

In general, Erlang is a pretty good language. I would never use it though because I think that functional programming is mostly a novelty, not for real world applications. I would find it highly unlikely to find any company start a project today in erlang. There are a lot fo quirks to the language which add to its cost, specifically, not using and, not really making it obvious how characters and strings work, for some reason the head of a character is the character code??? and the way the shell works. The readability is pretty good, especially with ligatures. I did notice that they do the >= =< comparison operators wrong. Very confusing... should be equal on the right in both cases, they broke my ligatures... Takes away from the writability.

Ratings

Readability: 7/10

Writability: 4/10

Ranking: 3/5

Javascript

Commentary

I really love JavaScript, especially functionally. This is normally the way i try tyo use JS. Whenever I write a program in JS, I tend to use map functions over for loops if I can. I find this helps me to think about moving data around in a better way. It is especially helpful when working with Promises, because I can use a map function to get asynchronous data from various sources, in parallel, essentially just firing off a ton of requests, the using the Promise.all function which takes an array of promises(the map function) I can wait for all the data to come back. I also took the time to get to know all of the fancy shorthands like the spread operator, and arrow functions, which after hearing you compare lambda functions to "anonymous" functions, helped me to feel more at home.

Google Searches

- js get charcode
- js from charcode

Caesar Implementation

```
const rotate = (letter, shift) => (/[a-zA-Z]/).test(letter) ?
String.fromCharCode((((letter.charCodeAt(0) - 65) + shift) % 26) + 65) :
letter;
const encrypt = (word, shift) => word.toUpperCase().split("").map(ltr =>
rotate(ltr, shift)).join("");
const decrypt = (word, shift) => encrypt(word, 26 - shift);
const solve = (word, maxShift) => [...Array(maxShift)].forEach((_, i) =>
console.log(`Shift: ${i}\t${decrypt(word, i)}`));

console.log(encrypt("ATTACK AT ONCE", 4))
console.log(decrypt("EXXEGO EX SRGI", 4))
solve("abcdeFGHIJKLmnopqrstuvwxyz ,?;{[()]}", 26)
```

Output

```
EXXEGO EX SRGI
ATTACK AT ONCE
Shift: 0 ABCDEFGHIJKLMNOPQRSTUVWXYZ ,?;{[()]}
Shift: 1 ZABCDEFGHIJKLMNOPQRSTUVWXY ,?;{[()]}
Shift: 2 YZABCDEFGHIJKLMNOPQRSTUVWX ,?;{[()]}
Shift: 3 XYZABCDEFGHIJKLMNOPQRSTUVW ,?;{[()]}
Shift: 4 WXYZABCDEFGHIJKLMNOPQRSTUV ,?;{[()]}
Shift: 5 VWXYZABCDEFGHIJKLMNOPQRSTU ,?;{[()]}
Shift: 6 UVWXYZABCDEFGHIJKLMNOPQRST ,?;{[()]}
Shift: 7 TUVWXYZABCDEFGHIJKLMNOPQRS ,?;{[()]}
```

```
Shift: 8 STUVWXYZABCDEFGHIJKLMNOPQR ,?;{[()]}
Shift: 9 RSTUVWXYZABCDEFGHIJKLMNOPQ ,?;{[()]}
Shift: 10 QRSTUVWXYZABCDEFGHIJKLMNOP ,?;{[()]}
Shift: 11 PQRSTUVWXYZABCDEFGHIJKLMNO ,?;{[()]}
Shift: 12 OPQRSTUVWXYZABCDEFGHIJKLMN ,?;{[()]}
Shift: 13 NOPQRSTUVWXYZABCDEFGHIJKLM ,?;{[()]}
Shift: 14 MNOPQRSTUVWXYZABCDEFGHIJKL ,?;{[()]}
Shift: 15 LMNOPQRSTUVWXYZABCDEFGHIJK ,?;{[()]}
Shift: 16 KLMNOPQRSTUVWXYZABCDEFGHIJ ,?;{[()]}
Shift: 17 JKLMNOPQRSTUVWXYZABCDEFGHI ,?;{[()]}
Shift: 18 IJKLMNOPQRSTUVWXYZABCDEFGH ,?;{[()]}
Shift: 19 HIJKLMNOPQRSTUVWXYZABCDEFG ,?;{[()]}
Shift: 20 GHIJKLMNOPQRSTUVWXYZABCDEF ,?;{[()]}
Shift: 21 FGHIJKLMNOPQRSTUVWXYZABCDE ,?;{[()]}
Shift: 22 EFGHIJKLMNOPQRSTUVWXYZABCD ,?;{[()]}
Shift: 23 DEFGHIJKLMNOPQRSTUVWXYZABC ,?;{[()]}
Shift: 24 CDEFGHIJKLMNOPQRSTUVWXYZAB ,?;{[()]}
Shift: 25 BCDEFGHIJKLMNOPQRSTUVWXYZA ,?;{[()]}
```

Log

Estimate: 30 minutes

Date	Hours Spent	Accomplishments
4/2	0.25	Write full program

Discrepancy of time

I underestimated my abilities at javascript. I think also functionally is the normal way I think about programming in javascript.

Overall Review

People love to hate on javascript but I find it the most applicable language by far in web development. It is really the only way to program a front-end which is a huge plus for the language and ensures its support for the years to come. The development community has to be one of the largest and the documentation is worlds ahead of anything else. I think this project of seeing other languages has only confirmed this for me. You can tell by the searches I made, that the language is well documented as I only needed to search for two things, and I found them on the first try. I will admit the program I wrote is a bit hacky, especially the map on the Array constructor. From what I've found it is the most functional way to do run something x times.

Ratings

Readability: 7/10

Writability: 8/10

Ranking: 1/5

Scala

Commentary

For this one, I didn't have to change to much. I started with my program from last time and reorganized the encrypt to be more like a functional manner. Decrypt didn't have to change at all, sweet. For solve, I googled if there was a .foreach function, and then when I learned it did, I looked up a way to generate a sequence. It easily gave me those answers given Scala's above average documentation and community and boom, I was done.

Google Searches

- scala .foreach
- scala generate sequence of int

Caesar Implementation

```
// run with scala caesar.scala
object Caesar extends App {

    println(encrypt("ATTACK AT ONCE", 4))
    println(decrypt("EXXEGO EX SRGI", 4))

    solve("abcdeFGHIJKLmnopqrstuvwxyz ,?;{[()]}", 26)

    def encrypt(input: String, shift: Int): String = {
        input
            .toUpperCase()
            .map(chr =>
                if (chr.isUpper)
                    ((chr - 65 + shift) % 26 + 65).toChar
                else
                    chr
            )
    }

    def decrypt(input: String, shift: Int): String = {
        encrypt(input, 26-shift)
    }

    def solve(input: String, maxShift: Int): Unit = {
        List.tabulate(maxShift)(_+0).foreach(i => println("Shift: " + i +
"\tResult: " + decrypt(input, i)))
    }
}
```

Output

```
EXXEGO EX SRGI
ATTACK AT ONCE
Shift: 0        Result: ABCDEFGHIJKLMNOPQRSTUVWXYZ ,?;{[()]}
Shift: 1        Result: ZABCDEFGHIJKLMNOPQRSTUVWXY ,?;{[()]}
Shift: 2        Result: YZABCDEFGHIJKLMNOPQRSTUVWX ,?;{[()]}
Shift: 3        Result: XYZABCDEFGHIJKLMNOPQRSTUVW ,?;{[()]}
Shift: 4        Result: WXYZABCDEFGHIJKLMNOPQRSTUV ,?;{[()]}
Shift: 5        Result: VWXYZABCDEFGHIJKLMNOPQRSTU ,?;{[()]}
Shift: 6        Result: UVWXYZABCDEFGHIJKLMNOPQRST ,?;{[()]}
Shift: 7        Result: TUVWXYZABCDEFGHIJKLMNOPQRS ,?;{[()]}
Shift: 8        Result: STUVWXYZABCDEFGHIJKLMNOPQR ,?;{[()]}
Shift: 9        Result: RSTUVWXYZABCDEFGHIJKLMNOPQ ,?;{[()]}
Shift: 10       Result: QRSTUVWXYZABCDEFGHIJKLMNOP ,?;{[()]}
Shift: 11       Result: PQRSTUVWXYZABCDEFGHIJKLMNO ,?;{[()]}
Shift: 12       Result: OPQRSTUVWXYZABCDEFGHIJKLMN ,?;{[()]}
Shift: 13       Result: NOPQRSTUVWXYZABCDEFGHIJKLM ,?;{[()]}
Shift: 14       Result: MNOPQRSTUVWXYZABCDEFGHIJKL ,?;{[()]}
Shift: 15       Result: LMNOPQRSTUVWXYZABCDEFGHIJK ,?;{[()]}
Shift: 16       Result: KLMNOPQRSTUVWXYZABCDEFGHIJ ,?;{[()]}
Shift: 17       Result: JKLMNOPQRSTUVWXYZABCDEFGHI ,?;{[()]}
Shift: 18       Result: IJKLMNOPQRSTUVWXYZABCDEFGH ,?;{[()]}
Shift: 19       Result: HIJKLMNOPQRSTUVWXYZABCDEFG ,?;{[()]}
Shift: 20       Result: GHIJKLMNOPQRSTUVWXYZABCDEF ,?;{[()]}
Shift: 21       Result: FGHIJKLMNOPQRSTUVWXYZABCDE ,?;{[()]}
Shift: 22       Result: EFGHIJKLMNOPQRSTUVWXYZABCD ,?;{[()]}
Shift: 23       Result: DEFGHIJKLMNOPQRSTUVWXYZABC ,?;{[()]}
Shift: 24       Result: CDEFGHIJKLMNOPQRSTUVWXYZAB ,?;{[()]}
Shift: 25       Result: BCDEFGHIJKLMNOPQRSTUVWXYZA ,?;{[()]}
```

Log

Estimate: 1 hour

Date	Hours Spent	Accomplishments
4/13	.25	Re-install scala (i'm a minimalist, sorry)
4/13	.25	Re-organize encrypt, decrypt, solve

Discrepancy of time

I was expecting to have to change the program more. I also didn't expect to have to re-install scala, I had thought I didn't uninstall it. Ultimately, I ended up saving a lot of time on this one.

Overall Review

Scala is a pretty good language overall. Functionally, it is a good choice also. Similarly, to javascript, I think it hits a good combination of being capable as both. Very rarely would the best way to design an application be purely functional or imperative. Working with arrays makes a lot of sense with map functions. Although, I prefer JS out of comfort, I have to give the objective

edge to scala here. The JS approach definitely involved some "hacky" techniques but that is kinda par for the course in a JS world. Scala gets the edge for readability and loses for writability because this program was much more verbose than the JS version I wrote.

Ratings

Readability: 8/10

Writability: 7/10

Ranking: 2/5

Rankings

1. JavaScript
2. Scala
3. ML
4. Erlang
5. Lisp