Harley Wakeman 1/29/23

Title: **Racket Assignment #1: Getting Acquainted with Racket/DrRacket + LEL Sentence Generation**

Abstract: During this Assignment, I was able to introduce myself to the programming language Racket, as well as the PDE DrRacket which obviously supports the Racket language.

Code for the LEL Sentence Generator:

```
#lang racket


;-----------------------------------------------
; LEL sentence generator, with helper PICK,
; serveral applications of APPEND, several
; applications of LIST, and one use of MAP
; with a LAMBDA function.
( define ( pick list )
( list-ref list ( random ( length list ) ) )
)
( define ( noun )
( list ( pick '( robot baby todler hat dog ) ) )
)
( define ( verb )
( list ( pick '( kissed hugged protected chased hornswoggled )))
)
( define ( article )
( list ( pick '( a the ) ) )
)
( define ( qualifier )
```

```
( pick '( ( howling ) ( talking ) ( dancing )
( barking ) ( happy ) ( laughing )
() () () () () ()
)
)
)
( define ( noun-phrase )
( append ( article ) ( qualifier ) ( noun ) )
)
( define ( sentence )
( append ( noun-phrase ) ( verb ) ( noun-phrase ) )
)
( define ( ds ) ; display a sentence
( map
( lambda ( w ) ( display w ) ( display " " ) )
( sentence )
)
( display "") ; an artificial something
)
```

Demo for the LEL Sentence Generator:

```
Welcome to DrRacket, version 8.7 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( pick '( red yellow blue ) )
```

```
'blue
> ( pick '( red yellow blue ) )
'red
> ( pick '( red yellow blue ) )
'red
> ( pick '( red yellow blue ) )
'red
> ( pick '( Racket Prolog Haskell Rust ) )
'Rust
> ( pick '( Racket Prolog Haskell Rust ) )
'Prolog
> ( pick '( Racket Prolog Haskell Rust ) )
'Rust
> ( pick '( Racket Prolog Haskell Rust ) )
'Prolog
> ( noun )
'(robot)
> ( noun )
'(robot)
> ( noun )
'(baby)
> ( noun )
'(baby)
>  ( verb )
'(protected)
```

> (verb)

'(kissed)

> (verb)

'(kissed)

> (verb)

'(hornswoggled)

> (article)

'(a)

> (article)

'(the)

> (article)

'(the)

> (article)

'(a)

> (qualifier)

'(howling)

> (qualifier)

'()

> (qualifier)

'()

> (qualifier)

'(happy)

> (qualifier)

'()

> (qualifier)

'()

> (qualifier)

'(talking)

> (qualifier)

'(howling)

> (qualifier)

'(dancing)

> (qualifier)

'(talking)

> (qualifier)

'(barking)

> (qualifier)

'(talking)

> (qualifier)

'(talking)

> (qualifier)

'(happy)

> (qualifier)

'()

> (qualifier)

'(howling)

> (noun-phrase)

'(the howling todler)

> (noun-phrase)

'(the robot)

> (noun-phrase)

'(a hat)

> (noun-phrase)

'(the dog)

> (noun-phrase)

'(a laughing robot)

> (noun-phrase)

'(the talking robot)

> (noun-phrase)

'(the howling baby)

> (noun-phrase)

'(the hat)

> (sentence)

'(the hat hornswoggled a dancing baby)

> (sentence)

'(the happy robot protected a talking todler)

> (sentence)

'(a dog hornswoggled the talking baby)

> (sentence)

'(a happy hat kissed a happy robot)

> (sentence)

'(the happy baby hornswoggled the dog)

> (sentence)

'(a dog protected the hat)

> (sentence)

'(a laughing robot chased a hat)

> (sentence)

'(the hat kissed a baby)

> (ds)

a hat hugged the baby

> (ds)

a dog hornswoggled a hat

> (ds)

a hat protected the robot

> (ds)

a dog kissed a robot

> (ds)

a talking dog protected the laughing hat

> (ds)

a happy todler kissed the howling baby

> (ds)

the happy dog hugged the hat

> (ds)

the dancing robot kissed the dancing dog

> (ds)

the dancing todler hugged a happy dog

> (ds)

a robot protected a dancing dog

> (ds)

the barking baby protected a baby

```
> ( ds )
the barking dog protected a happy robot
>
```