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

Cameron Francois
CSC 344
February 7, 2023

Abstract:

This programming assignment introduces us to the Racket programming language using the DrRacket program. This assignment provides us with an LEL sentence generator program that we are tasked to mimic the code. After following the code and mimicking the code we are then tasked to demo each thing a certain amount of times. This demonstrates how the code should function at its core and the different outputs that can be achieved with the same input.

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 8.3 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( pick '( red yellow blue ))
'red
> ( pick '( red yellow blue ))
'yellow
> ( pick '( red yellow blue ))
'blue
> ( pick '( red yellow blue ))
'yellow
> ( pick '( Racket Prolog Haskell Rust ))
'Haskell
> ( pick '( Racket Prolog Haskell Rust ))
'Prolog
> ( pick '( Racket Prolog Haskell Rust ))
'Racket
> ( pick '( Racket Prolog Haskell Rust ))
'Rust
> ( noun )
'(hat)
> ( noun )
'(hat)
> ( noun )
'(baby)
> ( noun )
'(baby)
> ( verb )
'(chased)
> ( verb )
'(hornswoggled)
> ( verb )
'(protected)
> ( verb )
'(hugged)
> ( article )
'(a)
> ( article )
'(the)
> ( article )
'(a)
> ( article )
'(a)
> ( qualifier )
'(dancing)
> ( qualifier )
'()
> ( qualifier )
'(howling)
> ( qualifier )
'()
> ( qualifier )
'(dancing)
> ( qualifier )
'(dancing)
> ( qualifier )
'()
> ( qualifier )
'()
> ( qualifier )
'()
> ( qualifier )
'()
> ( qualifier )
'()
> ( qualifier )
'(howling)
> ( qualifier )
'()
> ( qualifier )
'(laughing)
> ( qualifier )
'()
> ( qualifier )
'(dancing)
> ( noun-phrase )
'(a dancing dog)
> ( noun-phrase )
'(the barking hat)
> ( noun-phrase )
'(a todler)
> ( noun-phrase )
'(the hat)
> ( noun-phrase )
'(a happy robot)
> ( noun-phrase )
'(the howling dog)
> ( noun-phrase )
'(a happy todler)
> ( noun-phrase )
'(the dog)
```

```
> ( sentence )
'(the robot hugged a happy robot)
> ( sentence )
'(a laughing robot chased a robot)
> ( sentence )
'(the dog chased the baby)
> ( sentence )
'(the dancing todler chased a hat)
> ( ds )
a todler protected the todler
> ( ds )
a baby chased a todler
> ( ds )
a baby protected the dog
> ( ds )
the dog protected the baby
> ( ds )
the hat hugged a dancing baby
> ( ds )
a todler kissed the robot
> ( ds )
a todler kissed a laughing robot
> ( ds )
the dancing robot hugged a laughing dog
> ( ds )
a baby kissed the robot
> ( ds )
a robot kissed a dog
> ( ds )
the hat kissed the dancing todler
> ( ds )
a hat chased the barking dog
>
```