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

---

Abstract:

In this assignment, I studied and ran a program written in Racket in the DrRacket program development environment (PDE). I tried learning about the syntax and behavior of the language. For that, I ran a given program and generated a demo, which is provided below.

Code for LEL Sentence Generator:

```
#lang racket
;-----------------------------------------------
; LEL sentence generator, with helper PICK,
; several 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 toddler 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 LEL Sentence Generator:

```
Welcome to DrRacket, version 8.8 [cs].
Language: racket, with debugging; memory limit: 128 MB.
>  ( pick '( red yellow blue ) )
'blue
> ( pick '( red yellow blue ) )
'blue
> ( pick '( red yellow blue ) )
'yellow
> ( pick '( red yellow blue ) )
'blue
> ( pick '( Racket Prolog Haskell Rust ) )
'Prolog
> ( pick '( Racket Prolog Haskell Rust ) )
'Haskell
> ( pick '( Racket Prolog Haskell Rust ) )
'Rust
> ( pick '( Racket Prolog Haskell Rust ) )
'Racket
> (noun)
'(baby)
> (noun)
```

```
'(dog)
> (noun)
'(robot)
> (noun)
'(dog)
> (verb)
'(kissed)
> (verb)
'(hugged)
> (verb)
'(hugged)
> (verb)
'(kissed)
> (article)
'(a)
> (article)
'(a)
> (article)
'(the)
> (article)
'(a)
> (qualifier)
'()
> (qualifier)
'(dancing)
> (qualifier)
'(laughing)
> (qualifier)
'(happy)
> (qualifier)
'()
> (qualifier)
'(laughing)
> (qualifier)
'()
> (qualifier)
'(barking)
> (qualifier)
'(dancing)
> (qualifier)
```

```
'()
> (qualifier)
'()
> (qualifier)
'(talking)
> (qualifier)
'()
> (qualifier)
'()
> (qualifier)
'()
> (qualifier)
'(happy)
> ( noun-phrase )
'(a laughing robot)
> ( noun-phrase )
'(the happy hat)
> ( noun-phrase )
'(a happy toddler)
> ( noun-phrase )
'(a robot)
> ( noun-phrase )
'(the toddler)
> ( noun-phrase )
'(the happy dog)
> ( noun-phrase )
'(the talking robot)
> ( noun-phrase )
'(a laughing baby)
> ( sentence )
'(a howling baby hugged the toddler)
> ( sentence )
'(the hat protected the happy robot)
> ( sentence )
'(a barking baby hugged the baby)
> ( sentence )
'(the dog kissed a baby)
> ( sentence )
'(a happy baby hugged a baby)
> ( sentence )
```

```
'(a barking robot hugged the dog)
> ( sentence )
'(the robot protected a hat)
> ( sentence )
'(the robot chased a baby)
> ( ds )
the hat chased the laughing hat
> ( ds )
the talking hat chased the laughing dog
> ( ds )
the robot hugged a hat
> ( ds )
a baby kissed a laughing toddler
> ( ds )
the baby hugged the hat
> ( ds )
a laughing dog kissed a barking baby
> ( ds )
the talking baby chased a toddler
> ( ds )
the laughing toddler chased a hat
> ( ds )
the howling robot chased a talking dog
> ( ds )
the dancing robot hugged the toddler
> ( ds )
a laughing dog hornswoggled a talking hat
> ( ds )
a howling hat chased a dog
>
```