COMP 301 Fall'20
Project 2

Altay Atalay	64568
Ege Yelken	61742
Umur Demircioğlu	64609

Part A:

1. Syntax and Datatypes → *data-structures.rkt*
2. Values → *data-structures.rkt*
3. Environment → *environments.rkt*
4. Behavior Specification → *interp.rkt*
5. Behavior Implementation → *lang.rkt*

Part B:

```
(define init-env
        (lambda ()
          (extend-env
           'z (num-val 1)
             (extend-env
              'y (num-val 5)
                (extend-env
                 'x (num-val 10)
                   (empty-env)))))))

(define env (init-env))
```

env(x) = 10, env(y) = 5, env(z) = 1

[]env

[x=10]env

[y=5][x=2]env

[z=1][y=5][x=10]env

Part C:

Expressed Values: Int+Bool
Denoted Values: Int+Bool

Part D:

4- Lang.rkt: even?-exp

(expression ("even?" "(" number ")") even?-exp)

Interp.rkt : value-of: even?-exp

```
(even?-exp (num)
        (if (eq? 0 (modulo num 2))
            (bool-val #t)
            (bool-val #f)))
```

Test.rkt : even-tests

```
(even-test1 "even?(2)" #t)
(even-test2 "even?(5)" #f)
(even-test3 "even?(9)" #f)
(even-test4 "even?(16)" #t)
```

Syntax:

- Expression :: even?(Expression)
  even?-exp(num)

This expression checks whether the input is even or not. It takes a number as an argument and returns boolean.