Part A)

The components of a language are:

* **Text

  Description automatically generatedSyntax** is defined in lang.rkt file
* **Text

  Description automatically generatedValues** are defined in the first part of the data-structures.rkt file
* **Text

  Description automatically generatedEnvironment** is implemented in environments.rkt file
* **Behavior specification** can be specified in a format and in interp.rkt:

(value-of (if-exp exp1 exp2 conds exps exp3) p) =

{ (value-of exp2 p) if (expval -> (value-of exp1 p) =#t)

{ (value-of (car conds) p) if (expval -> (value-of (car exps) p) =#t)

{ (value-of (cadr conds) p) if (expval -> (value-of (cadr exps) p) =#t)

{ (value-of (caddr conds) p) if (expval -> (value-of (caddr exps) p) =#t) .

.

{ (value-of exp3 p) else

For string expressions:

(value-of (str-exp exp) p) =

p

* Text

  Description automatically generated**Behavior implementation** are implemented in the interp.rkt file

Text

Description automatically generatedPart B)

[x = 2]

[y = 3]

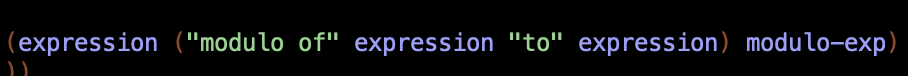
[z = 4] p

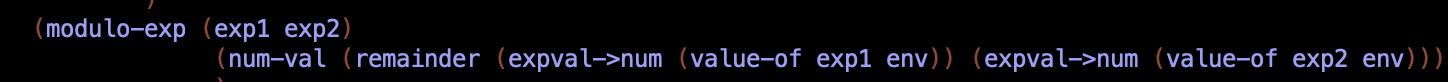
Part C)

ExpVal = Int + Bool + Str

DenVal = Int + Bool + Str

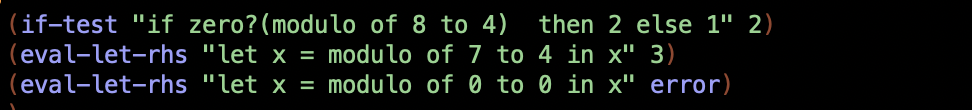
Part D)

 We created a modulo-exp which has a syntax:

 “modulo of expression to expression”

It takes two expressions, then we return num-val of the remainder of the number of expval of the first expression to the number of expval of the second expression with value-of method.

Part E)

Our test cases worked perfectly.

In this project both of us worked in every part of the questions we worked together and tried to answer each question by making brainstorming.

Bonus question is done.