**Part A.** This part will prepare you for the following parts of the project. (15 pts)

(1) Write the 5 components of the language :

* Syntax and datatypes
  + a-program → Program ::= Expression
  + Const-exp → Expression ::= Number
  + diff-exp → Expression ::= -(Expression, Expression)
  + zero?-exp → Expression ::= zero? (Expression)
  + if-exp → Expression ::= if Expression then Expression else Expression
  + var-exp → Expression ::= Identifier
  + let-exp → Expression ::= let Identifier = Expression in Expression
* Values
  + Expressed values: Possible values of expressions
  + Denoted values: Possible values of variables
* Environment
  + ranges over environments
  + [] denotes empt environment
  + denotes (extend-env var val )
  + abbreviates
  + denotes the environment where the value of var1 is val1, etc.
* Behavior specification
  + We specify the behavior of programs, expressions, and Observer(such as value-of)
* Behavior implementation
  + Scanning
  + Parsing
  + Evaluation

(2) For each component, specify where or which racket file (if it applies) we define and handle them.

* Syntax and datatypes
  + a-program → In **interp.rkt** file defined, and handled.
  + Const-exp → In **interp.rkt** file defined, and handled.
  + zero?-exp → In **interp.rkt** file defined, and handled.
  + if-exp → In **interp.rkt** file defined, and handled.
  + var-exp → In **interp.rkt** file defined, and handled.
  + let-exp → In **interp.rkt** file defined, and handled.
* Values
  + Expressed values: In **datastructures.rkt** file we define expressed values and it is either a number, a boolean, a string or a procval.
  + Denoted values: Not applicable
* Environment
  + In **environments.rkt** file first initial environment defined, then environment constructors and observerst implemented.
* Behavior specification
  + Grammatical specification is defined and handled in the **lang.rkt** file above the sllgen boilerplate comment.
* Behavior implementation
  + In the **lang.rkt** file , Scan&Parse implemented below the sllgen boilerplate comment.

**Part B.** In this part, you will create an initial environment for programs to run. (10 pts)

(1) Create an initial environment that contains 3 different variables (x, y, and z).

It is defined in “environments.rkt” and x,y,z initialized as 2,3,7. (init-env) = [x=2, y=3, z=7]

(2) Using the environment abbreviation shown in the lectures, write how the environment changes at each variable addition.

* + : after empty-env
  + : after (extend-env 'z (num-val 7)(empty-env))
  + : after (extend-env 'y (num-val 3)(extend-env 'z (num-val 7)(empty-env)))
  + : after (extend-env 'x (num-val 2)(extend-env 'y (num-val 3)(extend-env 'z (num-val 7)(empty-env))))
  + : final environment

**Part C.** Specify expressed and denoted values for MYLET language. (5 pts)

The used expressed values in MYLET language are either a number, a boolean, a string or a procval.

The used denoted values in MYLET language are symbols named as x, y, z.