Assignment-3 ppl

a. The role of the function value->lit-exp in the substitution model is to replace values with legal expression of the AST , if we don't use this function when doing substituted we will get illegal AST tree because the tree will contain values and not types of expressions.

b. The normal evaluator doesn't need the function value->lit-exp because when we want to substitute var-ref with value in this model we get the values as cexp which legal expression in the AST tree so we don't need to convert the value when doing substitute.

c. The environment model doesn't need value->lit-exp because in this model we don't do substitute of the values of the AST instead we hold the values in the environment and take it when we need them.

d. this program will have some parameter renamed more than ones,

```
(define x 1)

        ((lambda (x)(let ((x 2))

                (((lambda(x)(lambda(x)(+ x x x)))

                        x) 8)

                    ))

          x)
```

without renaming the function will return 24 , and with renaming the function will return the number 11.
The function with renaming :

```
(define x 1)

        ((lambda (y)(let ((x 2))

                (((lambda(z)(lambda(x)(+ x y z)))

                        x) 8)

                    ))

          x)
```

We can see that x renamed more than one, if we are not doing renaming every time that application computed the substitution of the variable will lead to capture of the free variable by the new binding.

e.

Advantages:

a. Derived expressions are independent of the semantic and the implementation of the core expressions.

b. Changing evaluation rule (if needed) will take place only on the rule itself. This will affect all the derived expression automatically.

c. We will have to make less tests to identify the type of the expression.

Disadvantages:

a. The derive implementation isn't as specific to the expression as if we would have write the expression as a core expression.

. b. Expression with derived expression will "spend" time in the derivation process

f. The main reason for switching from the substitution model to the environment model is that the last one is more efficient. The environment model is more efficient because we don't make the substitution and the renaming but matching to every procedure an environment with the values of the variables. Another reason is that we don't have to .evaluate values in distinct way

g. No, the apply-procedure get proc object that assumed to be prim-op or closure that was created by the function make-closure of the interpreter , none of those values is DrRacket closure even if we allow the interpreter to get such closure objects we don't have access to the closure parameters, body or environment in scheme because in scheme the primitive functions are like black box and we don't have way to pass those things into the interpreter

h. We can compute the type of letrec expression by the rules :

```
Typing rule:
  (letrec((p1 (lambda (x11 ... x1n1) body1)) ...) body)
  tenv-body = extend-tenv(p1=(t11*..*t1n1->t1)....; tenv)
;;   tenvi = extend-tenv(xi1=ti1,..,xini=tini; tenv-body)
;; If   type<body1>(tenv1) = t1
;;      ...
;;      type<bodyn>(tenvn) = tn
;;      type<body>(tenv-body) = t
;; then  type<(letrec((p1  (lambda  (x11  ...  x1n1)  body1))
...) body)>(tenv-body) = t

we  can  define  environment  for  each  body  and  evaluate  the
type  of  Ti  also  we  can  define  environment  that  we  check
the type of the letrec body.
```

9.

```
(let (( x 8))
      (let ((y x))
        (let
           ((z y))

          (let ((w z))

(+ x z y w)))))
```

Tw->Tz->Ty->Tx->Number