Question 1:

Yes. The interpreter of L3 need to rewrite the let expression into an application, so actually there is a rule before the syntactic form of let expression is evaluated.

Question 2:

The function solves the typing problem in the substitution model that occurs while activating a function. Before the activation, the function arguments are represented as values but the function body is an expression, so when the substitution occurs we change the values to their literal form (numExp, boolExp, strExp…) using the function valueToLitExp and solve the typing problem.

Question 3:

The valueToLitExp function is not needed in the normal evaluation strategy interpreter because arguments are not evaluated before they are passed to closures, so applyClosure gets CExps as arguments and replaces varRefs in the body with CExps.

Question 4:

The function valueToLitExp is not necessary because in the environment-model we don't use substitute so the typing problem doesn't occur.

Question 5:

We can switch to normal evaluation to reduce unnecessary computations in some cases. For example:

(lambda (a) (+ 1 2))

In this code section we don't need to value the argument.

Question 6:

We will switch to applicative evaluation to refrain from doing unnecessary computations. For Example:

(lambda (a) (* a a))

In this code section 'a' appears twice in the body so a normal evaluation would've evaluated it twice but in applicative evaluation it is only evaluated once.

Question 7:

a. In the substitution model we change only the free variables, so in a situation where all the term does not contain free variables, then every time we perform a placement operation within the function body of the parameters we will check if the variable is free, in our case is not ,so we will continue without execution "renaming". In such a situation there will be no confusion between the names of the variables because the variable is related and placed only with his vardecl.
   For example, ((lambda (x) (lambda (x) (+ 1 x)2)3) there are no free variables, so for the value x = 2 there is no free variable in the body (lambda (x) (+ 1 x)) so it will no substitute by any of the variables. Than, the value x = 2 will be place in the body (+ 1 x) so we don't need to worry for wrong placing.
b. In all non-evaluation actions procExp is done as we saw in class.
   For a naive replacement of procExp we will do the following:

- We will get a closure, a set of values and an environment.
- We will create a "Vars" variable that will accept the closing parameters.
- We will create a "litArgs" variable which will convert the set value to the litExp set.
- We will create a "body" variable that will accept the closing body
- We will return the evaluation of the alternative action with the variables we did

Question 8: