**Assignment 3 – Part 1**

1. A special form is a form that is evaluated in a different manner than the one we defined. **Let** is evaluated accordingly:

* The let variables have scope over the body of the let only.
* Each <vari> is bound to the value of each <expi>.
* All the bindings are performed simultaneously - they do not depend on each other.
* The initial values <expi> are computed outside the scope of the let.
* The evaluation of the let form involves the creation of closure value and its application to the initial values.

As we can see, the evaluation of **Let** is different from the non – special evaluation form, which is:

* Let (val0,…,valn)=(evaluate(exp0),…,evaluate(expn))
* If val0 is a primitive procedure: Apply the procedure val0 to the values (val1,…,valn).
* Else if val0 is a closure <Closure <p1...pn> <exp1>..<expk>):
  + Replace p1...pn by val1...valn in <exp1> .. <expk>
  + Evaluate the resulting expressions and return the value of <expk>.

Therefore, **Let** is of special form.

1. The purpose of the function **valueToLitExp** is to transform a value into a literal expression denoting this value. This function is needed in our interpreter as we try to evaluate procedures expressions of the language, using Applicative Order Evaluation with the substitution model. As we have learned, the value returned by a procedure is a closure, as the arguments given to it are Values and the body of it is a list of CExp expressions containing VarRef occurrences. Our objective in the evaluation of the procedure is to replace all VarRef occurrences in the body with the corresponding values of the arguments. But there is a problem with that, as VarRef are **expressions** and the arguments are **values**, hence we use the **valueToLitExp** function to map the values of the arguments to corresponding expressions, that will be replaced instead of the VarRef occurrences in the body to create a valid AST.
2. The **normal evaluation startegy** is similar to the applicative-eval. The only difference, between them, is that the step of argument evaluation occurs just before a primitive procedure is applied. Moreover, the normal order algorithm uses the **call by name** strategy of passing arguments to procedures without pre-computing them. Therefore, there is no need to substitute the value given in the procedure to an expression (with the function **valueToLitExp**) , because the values are added after the procedure was evaluated to a closure, which is a value.
3. The environment-model interpreter uses a data-structure called the environment, which binds every variable in the program to its value, depending on it’s scope. This model helps us to pre-process VarRef expressions in the body of the procedure so that when we evaluate the body, we will use the value of each VarRef that was bound to it in the environment, and thus make the VarRef substitution a lazy operation. Therefore, we won’t need to change the value given in the procedure to an expression (with the function **valueToLitExp**), because the values are added during the reduction of the body using the environment data structure.