TASK1

a) When we create our procedures and general structures in a way, we define them to be "call by value" we do not make changes on the original value. We create a clone of the value and store it separately, thus when we call the value with its reference from the memory after applying a procedure, we always get the initial value. Its advantage is that we will be able to store the original value itself even if we make changes on the value. Thus, when we make some undesired actions, we will be still able to reach the original value. However, its disadvantage is that when we want to make permanent changes on the value original reference shows we have to call "setref" operation each time
When we create our procedures and general structures in a way, we define them to be "call by reference" we do make changes on the original value. We will not create a clone value, thus when we make changes by giving the reference as parameter the original value will be affected. The advantage of "call by value is that" we do not have to re-assign new value to the original reference each time we want to make permanent changes, however this also means that when we make undesired actions, we will lose the initial value, this may cause major issues.

b) When we create our structure to make our language "call by name" we store the expressions that methods take in the forms of "thunks" for future evaluation when needed as "call by need". However, their one and major difference is that "call by name" evaluates the parameter stored in "thunks" each time the parameter appears in the body of the method. Thus, if the method changes any value that will cause difference in the evaluation of the expressions stored in "thunks", we may get different expvals each time we evaluate the expressions stored in "thunks" at the evaluation of the body of method. In "Call by need" thunks are evaluated at the first time it appears in the body of the method, thus even if the method changes any value that would cause difference in the evaluation of the expression stored in thunks after the first time the parameter seen in the body, the value of the expression does not change within the method. The advantage of "call by need" is that we do not have to re-evaluate the value of expression in the thunks each time we see them, thus it is computationally more efficient, but it also reduces the flexibility of the code. The advantages of "call by name" is that it's being more flexible, the changes made within the body of the method reflects the changes each time we re-evaluate the value of expressions in thunks. But it is also harder to trace and computationally more expensive.

Task 2

The statement starting with (var-exp) should be placed within the method "value-of" in the interp.scm. The statement starting with (value-of-thunk) should be placed in interp.scm. These pieces of codes are necessary in order to evaluate the value of the expressions stored in thunks and store the evaluated version in the store at the original location of thunks.

a) Value-of-thunk
   This function takes a thunk which is waiting for evaluation, then by passing the expression and environment stored in thunk to value-of function, it basicly evaluate the value of the expression in the thunk with the environment created at the same time with the creation of thunk. This process can be called defrosting.

b) (Var-exp)
   When we have a var-exp to evaluate in our hands we first get the location of the var in store, apply-env method returns the reference of the var in the store. Then with deref we get the value with the reference we get in the previous stage. Then we analyze it whether this value is a expval or a thunk which waits to be evaluated. If it is an expval we directly return it, however if it is a thunk we call value-of-thunk previously described. Then in order to store the value we get from value-of-thunk method to the original location thunk was stored; we use setref method.

   Thanks to these two methods we get the value stored in the thunks and store it at the original location where thunks were stored for present and future usage.

In this project workload is divided among the members equally according to the grading Sezen Zeynep Sumer completed the second part (task 4 and task 5) and Cem completed (task 1, task2 and task3).