Part 1:

1. The purpose of **valueToLitExp** is to convert from value (number, Boolean, primOp etc...) to expression because our AST built/work in the substitution model by expressions, without the usage of **valueToLitExp** the resulting body is not a valid AST.

2. Because there is no need to substitute the value before evaluating, it substitutes the expressions directly without transform to value, so we don't have error in types.

3. The two strategies for evaluating a let expression are normal-Evaluation and Applicative-Evaluation.

4. (a) Giving a function wrong argument (Type error)–(define square (lambda (x) (* x x)))-> (square #t)

   (b) Missing Arguments – ((lambda (y x) (* y x)) 2)

   (c) Free undefined variable – (* 5 x)

   (d) computing errors – (/ 8 0)

5. special forms cant extended or redefined and have a different evaluation behavior which it need special analyze during evaluation, for example if, let, define, lambda and quote. Although Primitive operators built-in basic operations in programming language that perform fundamental computations on primitive data type.

6. In substituting model processes we need to pass over all the AST each time when recursive calling

   For example lets define a fib(n) program:

```
(define fib (lambda (n)
    (if (or (= n 1) (= n 0)) 1
                (+ n (fib (- n 1))))))
```

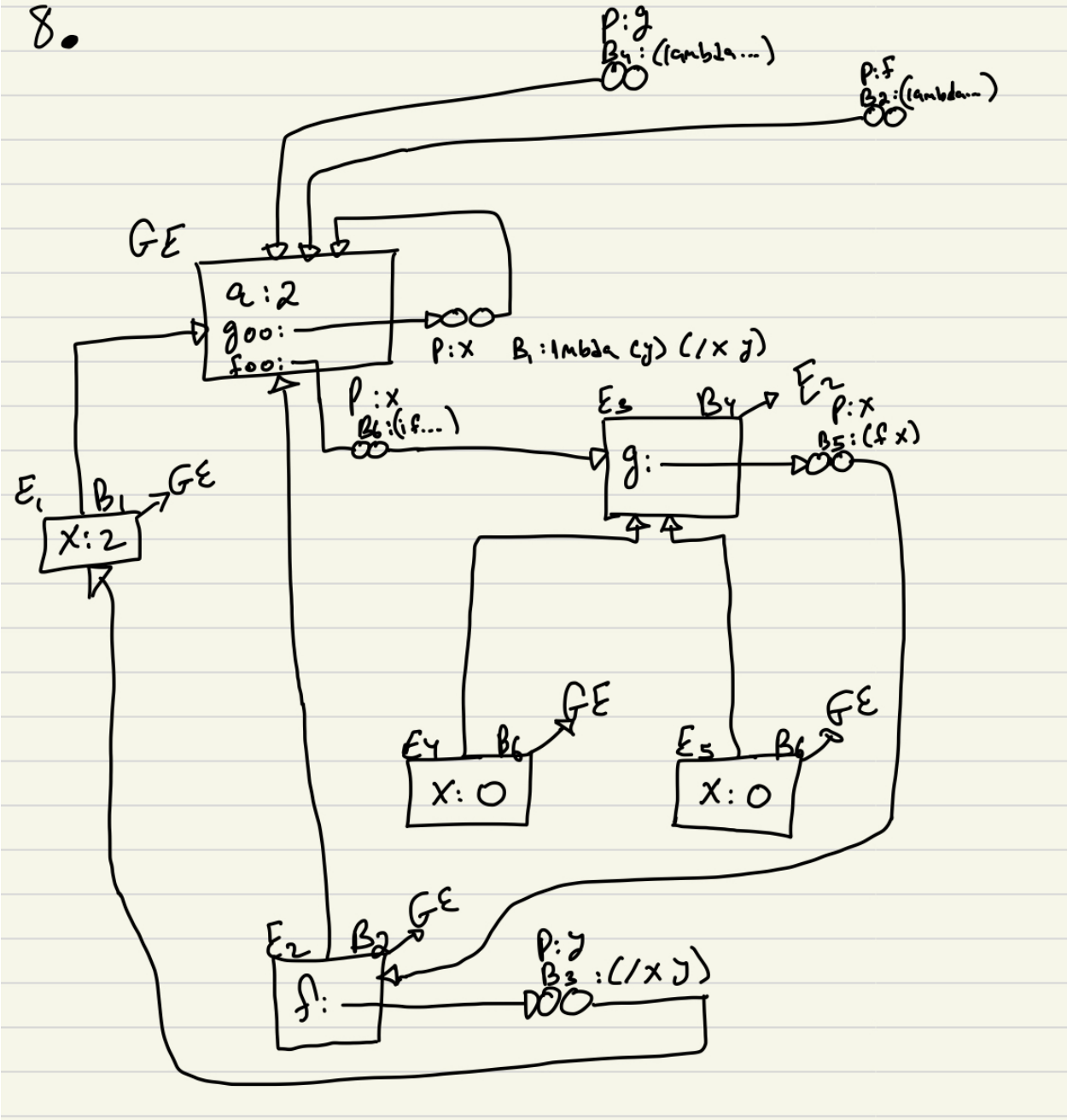
   When we evaluate fib(4) using substitution model in the recursive call we need to substitute n with needed to pass on the whole AST which is very expensive.

```
Fib (4) -> (lambda (4) (if (or (= 4 1) (= 4 0)) 1 (+ 4 (fib (- 4 1)))))
   (if #f 1 (+ 4  fib(3))
   (+ 4 fib(3))
Fib (3) -> (lambda (3) (if (or (= 3 1) (= 3 0)) 1 (+ 3 (fib (- 3 1)))))
   (if #f 1 (+ 3  fib(2))
   (+ 3 fib(2))
also for the rest computing.
```

In environment model we create for each procedure a sequence frames which represents a substitution of variables by values calling binding to use it multiply times without unnecessary calls like substitution model.

7. the main reason for implementing an environment using box is to enable mutation, which allows for changing of variable values. In the box environment model, variables are bound to boxes that contain values, and these values can be changed by accessing the box. This is needed to properly model recursion (letrec) and to model the global environment with forward usage of global variables and global mutual recursive procedure.

8.



2.1.3 Why is bound? expression has to be a spesial form, and cannot be a primitive or a user function?

'bound?' should be a special form because it will treat with variables in a different way than the regular forms so it cannot be a primitive because it is also depend on the way we define the environment and how to access to the first definition, inaddition it cannot be a user function because the lack of the ability of access to virable bindings information.

2.2.2 Can it be implemented as a user function, primitive or special form?

the expression that comes after 'time' is with type CExp so it can be Atomic or compound, therefore it cannot be implemented as primitive.

in contrast we can implement time as special form so we can treat with the ambiguous expression in different way than the regular forms