Implementation Document
Name:Rui Peng CDF:c5pengru
Name:Yuan Wang CDF:g3naiver

Your implementation of global name bindings (define):
Although there are no global variable in Haskell, we need to implement the Symbol Table for every time evaluate and return Expr with or without Symbol Table. Symbol table is a list, which contains several pairs, represents global variables. The first element of the pair is symbol, the second element is stored the value of this symbol (have not evaluated). If the keyword is "define", then need to pass symbol and value in the table and return the new Symbol Table. For some direct get NameError Boolean function, just return the solution. And For another function, just return the result of evaluate and the same table. When find a symbol during evaluation, first find it in the Symbol Table. If find it, then replace it,If not throw NameError.

Local name bindings let:
Let: similarly using a local symbol table, constructing as global Symbol Table. Using the function bind. For lexical scope, we must check the symbol in the local Symbol Table, then the global Symbol Table. If the symbol appears in any of table, replace it. If not throw NameError.

Let*: let* is almost same as let. Moreover, it can support (let ([x 10][y x])). So we update the local Symbol Table, in "let <expr>", <expr> part can also check local table, so I use the function binds, to insert into the local table recursively.

Local name binding functions:
there will be two types (define (f x) <body>) and (define g (lambda (x) <body>)). At first, we change the first type to the second, then implement the local bind which store in the global Symbol Table. When use this function, search from the global Symbol at first to find this function. Use the function Apply (line:497,504), to replace the local binding variable to the variable applied.

Lexical Scope:
Have already mentioned in let part. Check local Symbol Table first, then the global Symbol Table.

Implementation of static checking of names:
First check special symbols ("+", "*", "not", "equal?", "<", "list", "empty?", "first", "rest", "and", "or", "if", "cond", "let", "let*", "lambda", "define", "else"). These symbols should not be defined or it will throw NameError. Because every time pass global table in the evaluate function, so when a symbol appears in the Symbol Table, then get it value or Compound Part and evaluate it. If not just throw the Name error.