Final Project Writeup

I chose to extend my project to include lexical scoping. Ocaml itself implements lexical scoping, which manifests substitution semantics. The main difference between lexical and dynamic scoping is the difference in their environment application. In lexical scoping, the function is evaluated in the environment where the function was *defined*. In dynamic scoping, the function is evaluated in the environment where the function was *called*. Since eval_d and eval_l are very similar, I abstracted the common code out into a seperate function to avoid redundancy. My tests for eval_l and eval_s were very similar since they should produce the same results except when passing in a function as an expression.

This difference can be seen in this code snippet shown in class:

```
let x = 1 in let f = fun y = x + y in let x = 2 in f 3
```

In a dynamically scoped enviroment, the value of x would be 2 and the function would evaluate to 5. In a lexically scoped environment, the value of x would be 1 and the function would evaluate to 4.

Modifications from dynamic to lexical

1. Fun in lexical evaluates to a Closure of the function and the current environment.
2. App in lexical matches to a Closure instead of a function.
3. Letrec in lexical mutates the original unassigned environment to the evaluation of the first expression of letrec.