For my extensions:

1) I added the binary operators DIVIDE and GREATERTHAN. It was pretty easy to do this as I just had to update the binop type signatures. I then had to update my "binop_to_concrete_string" and "binop_to_abstract_string" functions to allow the program to print out the operations. Finally, because I only had one "binop_eval" function that I used in all "eval_" functions, I just had to update that function to handle dividing ints and checking which is greater.

2) I also modified the environment semantics to manifest lexical scope instead of dynamic scope. To do this, I wrote the "eval_l" function which was quite similar to the "eval_d" function. One of the differences was that for functions, I had to return a closure containing the expression and environment instead of just returning the expression as a value. The other difference was that for function application, the body of the function to be applied now had to be evaluated in the lexical environment from the closure.