- Read.c
    - Parse.c
    - Lexer.c
    - TypeChecker.c

- Lexer.c
    - (Side Note) Token is a struct having:
        - Type: what is it: (ex: operator, identifier, argument)
        - Value: what its assigned (ex: +, =, 1, {, })
    - Lexer
        - Takes in a string
        - Token list
        - String is the program this generates the tokenlist used by parse
    - AddToken
        - Takes in token and list, returns list with token added. Adds given token to given list
    - CreateToken
        - Takes in char, returns a token

- Parser.c
    - This class takes user input and creates a tree for type checker to validate. This tree is further used in Eval.c
    - Parse
        - Takes in tokenlist returns a cons object
    - Cons (Constructor)
        - Takes is a cons object from parse, returns a tree representing the code. This functions appends a type for each element is given tree.

- Type Checker.c
    - (Side note) Valid typecheck:
        - means that arguments match operators specifications.
            - i.e., + looking for int, double, etc.
        - Checks for log k errors
            - i.e., %, / by 0
        - Checks for cast statement
            - Int x = 1
            - double x = 3.4
    - Validate
        - Takes in tree from parse, returns true if code tree is valid, false otherwise
    - Checktype
        - Takes in an operator and its arguments, returns tree if arguments are valid typeof operator, false otherwise.
    - Typeof

        - Takes in operator, arguments. Compares argument to defend list of approved types by operator

- Eval.c
    - Eval
        - Takes in an expression and environment where environment is a hash-table that takes in a list expression and is the tree from Parse.c. Returns the tree after its been evaluated.
    - Print
        - Takes in tree from eval, returns nothing, prints the evaluated expressions from eval.
    - Apply
        - Takes in operator, argument(s), returns a tree with implemented functions using the tree given in eval we compare the operator with our defined functions “operators” and uses the arguments to compute the operation