We wrote three lido files for our specification, one for the Tiger grammar, one to specify the type checking and a third to specify name analysis.

The grammar parsing is complete, almost identical to what we turned in for the last milestone.

Name analysis is complete. The biggest challenge in name analysis was altering the tree structure to facilitate proper scoping rules, including joined scopes for 'let' expressions and 'for' loops implicitly defined nested scopes in 'let' expressions, and handling both of our namespaces. We tested our implementation with both good and erroneous test cases, and every one ran as expected.

Type analysis is very nearly complete. We worked our way through the features discussed in the 'Type Analysis Tasks' document, in order of complexity. We declared predefined types, handled type declaration and use, declared predefined operators, handled procedure definitions and calls, and handled declaration and use of user defined types. We skipped a few features unnecessary for Tiger; procedure and operator overloading, structural type equivalence, and type conversion. Unfortunately, there were a few features we were not able to complete; records and arrays are still showing a particularly difficult to solve problem, and the implicitly typed variable declaration syntax is still showing a cyclic dependency we have not been able to solve.

There was one problem we were working with that, on closer inspection of the language definition, turned out to be a non-issue. As it turns out, procedure declarations that do not specify a return type are not implicitly typed, but must have NO return type. Earlier we had been treating this as a serious cyclic dependency.

In the end, we now have a parser and analyzer for a not quite complete but self-sufficient subset of the Tiger specification. It's complete enough to provide a workable programming language and enough flexibility for the rest of the project.