# CIS4650 Checkpoint 2 Documentation

*Team: Christian Foote, Drew Mainprize*

**What has been completed:**

For this checkpoint we have completed work that was left incomplete at the end of checkpoint 1, as well as implemented the symbol table, and type checking.

We started work by getting to where we should have been at the end of checkpoint 1, fixing bugs and completing our abstract syntax tree, fixing small issues in a number of our grammar rules and actions.

With the improvements made during Checkpoint 2 semantic analysis can now be performed. The compiler can now recognize and differentiate different scope levels within a given C- compiler, as well as perform different kinds of type checking, including ensuring array indices are integers, return types match that of the function, or that assignment operations are valid.

**Techniques & Design:**

An incremental process was used across this checkpoint. Starting initially with all the work that needed to be done to get our compiler ready to start on checkpoint 2, we started by creating our Symbol Table as advised in the assignment description, using an array of hash tables as discussed in Lecture 8-Type Checking, so that we could easily track each individual scope. Once this was done we were able to move on to working on type checking and error recovery, slowly making our way through our test files writing and testing

**Lessons learned:**

This checkpoint started by emphasizing the importance of keeping up with the assigned work, as we had to start from behind and get our compiler up to a point where we could even start for checkpoint 2. This caused issues where our attention was split between what needed to be done for checkpoint 2, and what we had done for checkpoint 1, and with the split attention and dependence of one on the other, this slowed down our already difficult progress.

Through this, however, we have also been able to learn the importance of working within our individual strengths. We were faced with the issue I talk about above, and to deal with it we decided to split our attention, with one of us working on getting the first part working, and the other working on what was needed for checkpoint 2, as we each had one we were significantly more comfortable with.

**Assumptions & Limitations:**

For our current implementation of our compiler we have assumed that syntactic errors will not be tested, due to our time constraint, and the assignment description specifying that semantic errors will be tested.

**Potential Improvements:**

The first thing that could potentially be improved is the implementation of syntactic error checking.

We could also spend time cleaning up our grammar rules and actions in cminus.cup, as many of them have been changed many times since being written, and we have learnt a lot since writing them. Along with this there are times when the compiler will miss parts of the provided tests files, and this should be a high priority to fix if it begins to occur in our own test files.

Time could also be spent improving error reporting, implementing things such as improved, more detailed messages, as well as reporting both error line, and column.

**Individual Contributions:**

*Drew:*

- Symbol table
- Semantic Analyzer
- Symbols

*Christian:*

- Finished incomplete ShowTreeVisitor and cminus.cup
- Updated CM.java
- Writing documentation