

## Course Project – Due no later than 5 p.m. on May 7th

The purpose of the course project is to give you experience with processing language syntax, understanding the semantics of expressions in a language, and building the pragmatic mechanisms that implement the semantics of a language.

Because this is a semester-long project that represents 40% of the final course grade, there is a lot of risk in not making incremental progress on the project. For this reason, the project is broken down into three phases. Each phase represents incremental evolution of the language implementation. Phase 1 is the definition of the lexical and syntactic processing of the language. Phase 2 is the demonstration that your language processor can compile and execute basic language constructs. Phase 3 is the completed project. You should target allocating 2-3 weeks for each phase, so that you are not swamped with work at the end of the semester when there is no time and pressure is the most intense. I may require you to demonstrate the completion of Phases 1 & 2 at various points during the course to either the TA or myself to ensure that you are not stuck too long in a phase.

To implement *Tiger*, you may use Java, ML, Haskell or Scheme. For some of these languages, there are tools to assist you in automatically generating the lexical and syntax analysis parts of the interpreter, or you can do them by hand. In the case of Java, there is Jlex/JCup or ANTLR. In the case of ML, there is ML-Lex and ML-Yacc. For Haskell, you may use the monadic parser library that is available with GHC. Documentation for these tools/libraries is usually inadequate, so please allow yourself plenty of time to learn, experiment, and debug your parser and interpreter.

### Tiger Compiler/Interpreter Project

Tiger is a simple imperative language with a functional flavor that was defined by Dr. Andrew Appel at Princeton, and is often given as a course project in undergraduate compiler courses. The Tiger language definition is provided as a separate handout. There are two additions to the language description that are required for this project. You must implement a Boolean type with builtin constant values true and false and define the & and | operators to work for Booleans, not integers. You must also implement recursive function definition and execution. If you need to do so, you may introduce special syntax to denote a recursive function definition.

### Project Phase Submission Requirements.

For each phase of the project, you should submit, using the **turnin** program, a gzipped tar file of your project files that has a filename that includes your first initial and last name. For example, 'glavender.tar.gz' which should then un-compress and un-tar into a directory with the name 'glavender'. You must include a README.txt file that identifies which implementation language (and compiler/interpreter) was used for your Tiger interpreter, operating system and hardware platform on which you developed your project and clearly document the steps required to build your software from the source code. You must also include a 'makefile' that will build your project. For the final project submission, you must include documentation that defines the language or language subset that you have implemented and any limitations in your Tiger implementation.