Final Project Supporting Documentation

For the final project, I have decided to invent and implement an esoteric programming language. Through the language invention phase, the Frogger language was created. Frogger is a non-linear language with a built-in de-obfuscator. That is, Frogger code will not process in a top-down fashion and the language itself has trigger points for code obfuscation (See Language Definition.docx for more language-specific details).

Code Modules (all are pairs of .cpp and .h files unless otherwise noted):  
codeGenerationPhase – This phase generates the .cpp target code from the AST.  
compiler – A driver class for Frogger-to-C++ compilation.  
nodes – These files contain all node structures for the AST.  
obfuscator – These files process the obfuscation per specs in the Language Definition.  
parser – These files build the AST from the token stream provided by the scanner.  
phases.h – The superclass for all phases.  
scanner – These files create a token stream from the selected input file.  
stringConversionPhase – This phase converts string literal values from Frogger strings to C++ strings.  
summationPhase – This phase adds up all the ascii values for each line (used for control flow).  
tempAssignSubPhase – This phase generates the temporary variable declarations required.  
token – These files contain token definitions used by the scanner and parser.  
varDecSubPhase – This phase generates all variable declarations.

For Example input and output code, see the Samples folder and Samples->Output subfolder.