**Exercise 5.6.**

Straightforward modification of the while loop.

**Exercise 5.7.**

This caused some trouble because our parser initially was creating nodes incorrectly. Namely, the largest issue was to do with the initial variable declaration (or lack thereof) with reference to the AST. It appeared to declare the variable in the initializer, but it was not added in the context of the AST. Also, the final test (empty initialization, condition, and iteration) requires the break() statement which is not yet implemented. It wasn’t attempted, but our idea is this statement merely POPs off the stack until... something.

**5.11.**

Simple modification of the if-else statement structure.

**5.12.**

Initially we thought it was just a small modification of the &&’s if statement to ifFalse, but testing later showed dual true or false statements were swapped. After trying for a while and swapping true/false values in codegen, it seems like working (compiled similarly to the &&’s codegen).

**5.15** - Throw (bonus) (working in all the tests)

This appeared to be simple, but problems quickly arose. The analyze method just required a call to the internal JExpression field’s analyze. The codegen was trickier. The codegen call to the JExpression field’s own codegen was clear, but output wasn’t generating the thrown expression when the test file was compiled through j-- and ran. So we tracked the code generation in CLEmitter and found that the athrow instruction was considered a MISC operation, and appended a call in JThrowStatement’s codegen to reflect that, completing the required work. Unfortunately, this caused a pop off an empty stack. This problem was not resolved.

**5.21** - Primitive Long (bonus) (not fully working)

JLiteralLong was implemented very similarly to JLiteralInt. Like JLiteralInt, we found that we could call addLDCInstruction(), as it already had a case for if the parameter was a long. We also went through the code within Parser and other parts of the program in order to retrofit it so that the program not only accepted char, int, and boolean, but long as well. Unfortunately, we seemed to run into an error where longs were not being pushed onto the stack correctly by codegen. The root of this seemed to be deeply buried within CLEmitter, and as such, we were unable to figure out the problem.