Milestone 3

We've decided to code generate Jasmin IR. Part of it is because we intend to use the peephole optimizer as part of our GoLite project. One of the main challenges we've found generating lower-level target language (bytecode in this case) versus higher level languages is the need to use additional data structures at code generation time in order to keep track of the locals registers. For instance, in bytecode, upon declaring and assigning an integer "a = 5", we must remember its location on the JVM .locals register in order to "iload" it to the operand stack whenever it is used in expressions. Furthermore, we must compute the size limit of the .locals register and operand .stack for each function stack frame. We're planning to compute the .locals size limit in the type-checking phase as it would be just a matter of counting to number of local variables in the symbol table and we'll pass this number to the AST node for function declaration.

We now know most of the common key patterns to generate GoLite to Jasmin from testing and playing with the Jasmin compiler. However, we have yet to find a simple implementation to the GoLite "struct" in Jasmin. At first, we thought that a Struct is similar to a Java Class, however, upon testing this implementation, we've found that each Class necessitates its own bytecode file as it is in Java. Thus, if we used bytecode Classes as structs, compiling a GoLite with multiple structs would compile to multiple bytecode files which is not the best.

<u>Function Declarations</u>

- func functionName () { } has code template:
```
.method public static main([Ljava/lang/String;)V
                    return
.end method
```

<u>Expressions</u>

expr has code template: expr

- Arithmetic Operators like "+" , "-" , "*" , "/" , "%"..etc. Below is an example of the addition expression.
  exp1 op exp2 has code template:
  ```
  exp1
  exp2
  iadd
  ```
  (depends on the type of the resulting expression and op used)

- Comparison Operators like "&&" / "||" / "<" / ">" / "<=" / "==" / ">=" ..etc. Below is an example of the equality expression.
  exp1 op exp2 has code template:
  ```
  exp1
  exp2
  If_icmpeq true
  ldc_int 0
  goto stop
  true: ldc_int 1
  stop:
  ```

<u>Variable Declarations</u>

- Variable declaration without an assignment. Below is an example of integer variable declaration.
  Var varName varType has code template:
  ```
  bipush 0
  iistore 1 (location in locals)
  ```
- Variable declaration with an assignment. Below is an example of float variable declaration.
  Var varName varType = value has code template:
  ```
  bipush value
  fstore 2 (variable location)
  ```