EECS 4302 Winter 2020
Project Milestone 3
Team: Abdul Adeshina, Abdullah Basulaib, Chidi Okongwu, Shiyi Du

(Note: All the described features are illustrated in ***input-1.txt*** *file. We have 13 other files with more detailed illustration.)*
Our language is designed similar to java, each file can have multiple classes and all code logic is implemented within a class body,

Class Body: class *ID* { ... } //eg Class Account{ }

within the class body, there are 4 higher level constructs that are implemented

Declarations:
declare{
 Type ID = Value; //variable initialization, eg, int pi = 3.14;
 Type ID; …; //variable declaration, eg, int x;
}
Type can be int, bool or string
All variables used in the class are declared here. Declaration fields consist only of variable declarations, or variable initialization, variables can be accessed within class.

Assumptions: assume{ Boolean Expression1; …; }
Assumptions are used to tell the verification tool that it need not verify these cases where the assumption is false, you can think of it as a domain restricting function.

Functions: func *type function_name* (*type* parameter1, *type* parameter2...){
 require { *Boolean Expression1*; }
 functionBody;
 ensure { *Boolean* Expression1; }
}
You can have conditionals in function body just like Java:
 If (*Boolean Expression*) {...}
 Else if (*Boolean Expression*) {...}
 Else {...}

Functions have a return type, parameters, a precondition block, function body and post-condition block

Assertions: assert { Boolean expression; }
Assertions are used in a similar manner as class invariants and are used to ensure certain values or expressions hold throughout the class

The bodies of assertion, assumptions, preconditions and postconditions consist of expressions which evaluate to booleans, more than one expressions are considered conjunctions

Binary operator precedence from higher to lower
 * / + - > < >= <= == != && || => <=>