The input files illustrate the features that we have managed to implement until Milestone 3 and are sure that the alloy translation is accurate.

The features are the following:

* Variable declarations
  + Variables can be of type int or bool
  + Variables can be initialized to constant values
  + Variables can be initialized to the value of another (previously declared) variable’s value (of the same type)
  + Variables can be declared without assigning a specific value. In which case they will be assigned to their default values (0 for integer types, false for Boolean types)
* Variable assignment
  + Assignment must be type correct otherwise the compiler will report errors
* Variable references
  + Variables must be declared before referencing
* Arithmetic, relational, and **logical** expressions
* If statements (with contracts)
  + If statement can be with or without contracts but the compiler will only output alloy translation for if statements with contracts (otherwise there is nothing much to verify)
  + If statements can have zero or more else if statements and can have zero or one else statements
  + The syntax of the if statement with contracts is as follows

if\_require(expression)

if(expression){

// one or more assignments

} else if (expression) {

// one or more assignments

} else {

// one or more assignments

}

if\_ensure(expression)

* + As of milestone 3 deadline we don’t support nested if statements yet.
  + The bodies of if/else if/else statements must not have reassignment of one variable more than once (in each boy a single variable can be assigned only once). During milestone 2 meeting we discussed this issue with Professor Jackie and have been working on fixing it. However, supporting this task with the current output format results in very complicated output and we are working on new Alloy output format that will make this task possible.