Program -> **SourceElement***
SourceElement -> **Statement** | **MachineDeclaration** | **WhenStatement** | **Function**

Statement -> **RepeatStatement** | **DisplayStatement** | **MoveStatement** |
SetStatement | **SleepStatement** | **IfStatement** |
VariableDeclaration

MachineDeclaration -> define id = **Machinery**
Machinery -> analogPin[**Expression**] | digitalPinsOut[**Expression, Expression**] |
digitalPinsIn[**Expression, Expression**] | servo[**Expression**] | motor[**Expression**]

WhenStatement -> when {start | **WhenCondition**} **BlockStatement**
WhenCondition -> **WhenOrExpression**
Function -> func id(**Parameters**)
Parameters -> variable id {, variable id}$^{*}_{opt}$

SetStatement -> set id **Expression**
VariableDeclaration -> variable id = **AssignmentExpression**

IfStatement -> if **Expression BlockStatement** {else **BlockStatement**}$_{opt}$

WhenOrExpression -> **WhenAndExpression** { or **WhenAndExpression**}$^{*}_{opt}$
WhenAndExpression -> **EventExpression** { and **EventExpression**}$^{*}_{opt}$
EventExpression -> **ChangesExpression** | **EqualityWhenExpression** |
BetweenExpression
ChangesExpression -> id changes
EqualityWhenExpression -> EqualityExpression
BetweenExpression -> Expression between **Expression** and **Expression**

Expression -> AssignmentExpression {, AssignmentExpression}
AssignmentExpression -> ConditionalExpression {= AssignmentExpression}
ConditionalExpression -> LogicalORExpression
LogicalORExpression -> **LogicalANDExpression** { or **LogicalANDExpresion** }$^{*}_{opt}$
LogicalANDExpression -> **EqualityExpression** { and **EqualityExpression** }$^{*}_{opt}$
EqualityExpression -> **RelationalExpression** { **eqOp RelationalExpression** }$_{opt}$
RelationalExpression -> **AdditiveExpression** { **relOp AdditiveExpression** }$_{opt}$
AdditiveExpression -> **MultiplicativeExpression** {**addOp**
MultiplicativeExpression}$_{opt}$
MultiplicativeExpression -> **UnaryExpression** { **multOp UnaryExpression** }$_{opt}$
UnaryExpression -> {**unaryOP**}$_{opt}$ **LeftHandSideExpression**
LeftHandSideExpression -> **CallExpression**
CallExpression -> GetExpression | PrimaryExpression
GetExpression -> get id

PrimaryExpression -> (expression) | string | number | id

SleepStatement -> sleep **Expression**
BlockStatement -> { **Statement*** }
RepeatStatement -> repeat {**Expression** times}$_{opt}$ **BlockStatement**
MoveStatement -> **ForwardStatement** | **BackwardStatement** | **LeftStatement** | **RightStatement**
forwardStatement -> move forward **Expression**
backwardStatement -> move backward **Expression**
/* Questionable. Should we let students define what a "turn" is ? */
leftStatement -> turn left
rightStatement -> turn right
displayStatement -> display **Expression**

AddOp -> + | -
MultOp -> * | /
EqOp -> = | !=
RelOp -> < | > | >= | <=
UnaryOP -> not