\*parser,fgmstvx MACC\_N\_CHEESE Grammar 2016

\*tokens

"$eof$" EOF\_SYM.

"bool" BOOL\_SYM.

"break" BREAK\_SYM.

"case" CASE\_SYM.

"cheese" CHEESE\_SYM.

"decs" DECS\_SYM.

"do" DO\_SYM.

"else" ELSE\_SYM.

"end" END\_SYM.

"false" FALSE\_SYM.

"float" FLOAT\_SYM.

"for" FOR\_SYM.

"hiphip" HIPHIP\_SYM.

"if" IF\_SYM.

"int" INT\_SYM.

"listen" LISTEN\_SYM.

"otherwise" OTHERWISE\_SYM.

"select" SELECT\_SYM.

"shout" SHOUT\_SYM.

"then" THEN\_SYM.

"true" TRUE\_SYM.

"while" WHILE\_SYM.

"until" UNTIL\_SYM.

"[" LSTAPLE.

"]" RSTAPLE.

"(" LBANANA.

")" RBANANA.

"{" LMUSTACHE.

"}" RMUSTACHE.

":" COLON.

";" SEMICOLON.

"," COMMA.

"=" ASSIGN\_OP.

"+" PLUS\_OP.

"-" MINUS\_OP.

"\*" MULT\_OP.

"/" DIV\_OP.

"<" LT\_OP.

"<=" LE\_OP.

">" GT\_OP.

">=" GE\_OP.

"==" EQ\_OP1.

"!!" EQ\_OP2.

"!=" NE\_OP.

"ID" ID.

"INT\_LIT" INT\_LIT.

"FLOAT\_LIT" FLOAT\_LIT.

"CHEESE\_LIT" CHEESE\_LIT.

\*grammar

<program> <stmt\_list>

<stmt\_list> <statement> <stmt\_tail>

<stmt\_tail> <statement> <stmt\_tail>

| ""

<statement> <simple\_stmt>

| <struct\_stmt>

<simple\_stmt> <assign\_stmt>

| <listen\_stmt>

| <shout\_stmt>

| <break\_stmt>

<assign\_stmt> <variable> "=" <assign\_tail> ";"

<assign\_tail> <expression>

| "{" <init\_list> "}"

<listen\_stmt> "listen" <var\_list> ";"

<var\_list> <variable> <var\_list\_tail>

<var\_list\_tail> "," <variable> <var\_list\_tail>

| ""

<variable> "ID" <variable\_tail>

<variable\_tail> "[" <expression> "]"

| ""

<shout\_stmt> "shout" <item\_list> ";"

<item\_list> <expression> <item\_list\_tail>

<item\_list\_tail> "," <expression> <item\_list\_tail>

| ""

<break\_stmt> "break" ";"

<struct\_stmt> <if\_stmt>

| <loop\_stmt>

| <while\_stmt>

| <for\_stmt>

| <select\_stmt>

| <declaration>

<if\_stmt> "if" "(" <condition> ")" <stmt\_list> <else\_clause> "end"

<else\_clause> "else" <stmt\_list>

| ""

<loop\_stmt> "do" <stmt\_list> "until" "(" <condition> ")" ";"

<while\_stmt> "while" "(" <condition> ")" <stmt\_list> "end"

<for\_stmt> "for" "(" <for\_assign> ";" <condition> ";" <for\_assign> ")" <stmt\_list> "end"

<for\_assign> <variable> "=" <expression>

<select\_stmt> "select" "(" <condition> ")" <case\_list> "otherwise" ":" <stmt\_list> "end"

<case\_list> <case> <case\_list\_tail>

<case\_list\_tail> <case> <case\_list\_tail>

| ""

<case> "case" <int\_list> ":" <stmt\_list>

<int\_list> "INT\_LIT" <int\_list\_tail>

<int\_list\_tail> "," "INT\_LIT" <int\_list\_tail>

| ""

<condition> <expression> <cond\_tail>

<cond\_tail> <rel\_op> <expression>

| ""

<expression> <factor> <expr\_tail>

<expr\_tail> <add\_op> <factor> <expr\_tail>

| ""

<factor> <primary> <factor\_tail>

<factor\_tail> <mult\_op> <primary> <factor\_tail>

| ""

<primary> <literal>

| <variable>

| "(" <expression> ")"

<type> "bool"

| "int"

| "float"

| <cheese\_type>

<cheese\_type> "cheese" <cheese\_type\_tail>

<cheese\_type\_tail> "[" "INT\_LIT" "]"

| ""

<bool\_lit> "false"

| "true"

<literal> <bool\_lit>

| "INT\_LIT"

| "FLOAT\_LIT"

| "CHEESE\_LIT"

<mult\_op> "\*"

| "/"

<add\_op> "+"

| "-"

<rel\_op> "<"

| "<="

| ">"

| ">="

| "=="

| "!!"

| "!="

<var\_decs> "decs" "{" <dec\_list> "}"

<dec\_list> <declaration> <dec\_tail>

<dec\_tail> <declaration> <dec\_tail>

| ""

<declaration> <type> ":" <var\_dec\_list> ";"

| "hiphip" "[" "INT\_LIT" "]" <type> <var\_dec\_list> ";"

<var\_dec\_list> "ID" <var\_dec\_tail>

<var\_dec\_tail> "," "ID" <var\_dec\_tail>

| ""

<init\_list> <expression> <init\_tail>

<init\_tail> "," <expression> <init\_tail>

| ""