```
declaration_list : declaration_list declaration                  {$$ = addSibling($1, $2); }
| error                                                          {$$ = NULL; }; //EE

var_declaration : type_specifier var_decl_list ';'               {$$ = $2; setType($2, $1, false);}
| type_specifier error ';'                                       {$$ = NULL; yyerrok; }; //EE

scoped_var_declaration : STATIC type_specifier var_decl_list ';' {$$ = $3; setType($3, $2, true); }
| type_specifier error ';'                                       { $$=NULL; yyerrok; }; //EE

var_decl_list : var_decl_list ',' var_decl_initialize            {$$ = addSibling($1, $3); yyerrok;}
| error ',' var_decl_initialize                                  {$$ = NULL; yyerrok; }; //EE

var_decl_initialize : var_decl_id                                {$$ = $1;}
| error ':' simple_expression                                    { $$ = NULL; yyerrok; }; //EE
| var_decl_id ':' error                                          { $$ = NULL; }; //EE

var_decl_id : ID                                                 {$$ = newDeclNode(VarK, UndefinedType, $1);
| error '[' NUMCONST ']'                                         {$$ = NULL; yyerrok; }; //EE
| ID '[' error                                                   {$$ = NULL; }; //EE

fun_declaration : type_specifier ID '(' params ')' statement     { $$ = newDeclNode(FuncK, $1, $2);
| type_specifier error '(' params ')' statement                  { $$ = NULL; yyerrok; }; //EE
| type_specifier ID '(' error                                    { $$ = NULL; }; //EE
| ID '(' error ')' statement                                     { $$ = NULL; yyerrok; }; //EE
| error ')' statement                                            { $$ = NULL; yyerrok; }; //EE

param_list : param_list ';' param_type_list                      {$$ = addSibling($1, $3); yyerrok;}
| error ';' param_type_list                                      {$$ = NULL; yyerrok; }; //EE

param_type_list : type_specifier param_id_list                   {$$ = $2; setType($2, $1, false); }
| type_specifier error                                           {$$ = NULL; }; //EE

param_id_list : param_id_list ',' param_id                       {$$ = addSibling($1, $3); yyerrok;}
| error ',' param_id                                             {$$ = NULL; yyerrok; }; //EE
| param_id_list ',' error                                        {$$ = NULL; yyerrok; }; //EE

param_id : ID                                                    {$$ = newDeclNode(ParamK, UndefinedType, $1);
| error '[' NUMCONST ']'                                         {$$ = NULL; yyerrok; }; //EE
| ID '[' error                                                   {$$ = NULL; }; //EE

compound_stmt : '{' local_declarations statement_list '}'        { yyerrok;
| error '}'                                                      {$$ = NULL; yyerrok; }; //EE
| '{' error statement_list '}'                                   {$$ = NULL; yyerrok; }; //EE

statement_list : statement_list statement                        {$$ = addSibling($1, $2); }
```

```
 | statement_list error                                              {$$ = NULL; }; //EE

 statement : matched                                                 {$$ = $1;}
 | matched error                                                     {$$ = $1;}; //EE
 | unmatched error                                                   {$$ = $1;}; //EE

 matched : IF '(' simple_expression ')' matched ELSE matched         {$$ = newStmtNode(IfK, $1);
 | IF '(' error                                                      {$$ = NULL; }; //EE
 | error ')' matched ELSE matched                                    {$$ = NULL; yyerrok; }; //EE
 | WHILE '(' error                                                   {$$ = NULL; }; //EE
 | FOREACH '(' error                                                 {$$ = NULL; }; //EE

 unmatched : IF '(' simple_expression ')' statement                  {$$ = newStmtNode(IfK, $1);
 | IF '(' simple_expression ')' error ELSE unmatched                 {$$ = NULL; yyerrok; }; //EE
 | IF '(' error                                                      {$$ = NULL; }; //EE
 | WHILE '(' error                                                   {$$ = NULL; }; //EE
 | FOREACH '(' error                                                 {$$ = NULL; }; //EE
 | error ')' statement                                               {$$ = NULL; yyerrok; }; //EE

 expression_stmt : expression ';'                                    {$$ = $1; yyerrok;}
 | error ';'                                                         {$$ = NULL; yyerrok; }; //EE

 return_stmt : RETURN ';'                                            {$$ = newStmtNode(ReturnK, $1); yyerrok;}
 | RETURN error                                                      {$$ = NULL; }; //EE

 expression : mutable assignop expression                            {$$ = newExpNode(AssignK, $2);
 | error assignop expression                                         {$$ = NULL; yyerrok; }; //EE
 | mutable assignop error                                            {$$ = NULL; }; //EE
 | error INC                                                         {$$ = NULL; yyerrok; }; //EE
 | error DEC                                                         {$$ = NULL; yyerrok; }; //EE
 | mutable error expression                                          {$$ = NULL; yyerrok; }; //EE

 simple_expression : simple_expression OR or_expression              {$$ = newExpNode(OpK, $2);
 | error OR or_expression                                            {$$ = NULL; yyerrok; }; //EE
 | simple_expression OR error                                        {$$ = NULL; }; //EE

 or_expression : or_expression AND unary_rel_expression              {$$ = newExpNode(OpK, $2);
 | error AND unary_rel_expression                                    {$$ = NULL; yyerrok; }; //EE
 | or_expression AND error                                           {$$ = NULL; }; //EE

 unary_rel_expression : NOT unary_rel_expression                     {$$ = newExpNode(OpK, $1);
 | NOT error                                                         {$$ = NULL; }; //EE

 rel_expression : additive_expression relop additive_expression      {$$ = newExpNode(OpK, $2);
 | error relop additive_expression                                   {$$ = NULL; yyerrok; }; //EE
```

```
 | additive_expression relop error                                  {$$ = NULL; }; //EE

 additive_expression : additive_expression sumop term              {$$ = newExpNode(OpK, $2);
 | error sumop term                                                {$$ = NULL; yyerrok; }; //EE
 | additive_expression sumop error                                 {$$ = NULL; }; //EE

 term : term mulop unary_expression                                {$$ = newExpNode(OpK, $2);
 | error mulop unary_expression                                    {$$ = NULL; yyerrok; }; //EE
 | term mulop error                                                {$$ = NULL; }; //EE

 unary_expression : unaryop unary_expression                       {$$ = newExpNode(OpK, $1);
 | unaryop error                                                   {$$ = NULL; }; //EE

 mutable : ID                                                      {$$ = newExpNode(IdK, $1);
 | ID '[' error                                                    {$$ = NULL; }; //EE
 | error '['                                                       {$$ = NULL; yyerrok; }; //EE

 immutable : '(' expression ')'                                    {$$ = $2; yyerrok;}
 | '(' error                                                       {$$ = NULL; }; //EE
 | error ')'                                                       {$$ = NULL; yyerrok; }; //EE

 call : ID '(' args ')'                                            {yyerrok; $$ = newExpNode(CallK, $1);
 | error '(' args ')'                                              {$$ = NULL; yyerrok; }; //EE
 | ID '(' error                                                    {$$ = NULL; }; //EE

 arg_list : arg_list ',' expression                                {$$ = addSibling($1, $3); yyerrok; }
 | error ',' expression                                            {$$ = NULL; yyerrok; }; //EE


 ---------------------------------------------------------------------------
 These are the locations of the lines that have yyerrok macros that are not productions with
 an error token in them.   Be sure to add these yyerrok macros to your code as well.


 var_declaration : type_specifier var_decl_list ';' {$$ = $2; setType($2, $1, false);  yyerrok; }
 scoped_var_declaration : STATIC type_specifier var_decl_list ';' {$$ = $3; setType($3, $2, true); yyerrok; }
            | type_specifier var_decl_list ';' {$$ = $2; setType($2, $1, false); yyerrok; }
 var_decl_list : var_decl_list ',' var_decl_initialize {$$ = addSibling($1, $3); yyerrok; }
 param_id_list : param_id_list ',' param_id  {$$ = addSibling($1, $3); yyerrok; }
 expression_stmt : expression ';'        {$$ = $1; yyerrok; }
            | ';'                    {$$ = NULL; yyerrok; }
 immutable : '(' expression ')'             {$$ = $2; yyerrok; }
 arg_list : arg_list ',' expression      {$$ = addSibling($1, $3); yyerrok; }
```