## SEMANTIC RULES | GROUP\_34

	Grammar Production Rules			
No.	Top Down	Bottom Up		
1	<pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre>			
		<pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre>		
		free( <moduledeclarations>)</moduledeclarations>		
		free( <othermodules(1)>)</othermodules(1)>		
		free( <drivermodule>)</drivermodule>		
		free( <othermodules(2)></othermodules(2)>		
2	<moduledeclarations> → <moduledeclaration><moduledeclarations></moduledeclarations></moduledeclaration></moduledeclarations>			
		<moduledeclarations>.addr=createNewNode(label:moduleDeclaration,<moduledeclaration>.addr)</moduledeclaration></moduledeclarations>		
		<moduledeclarations>.syn_addr=addToStart(<moduledeclaration>.addr,<moduledeclarations>.syn_addr)</moduledeclarations></moduledeclaration></moduledeclarations>		
		free( <moduledeclaration>)</moduledeclaration>		
		free( <moduledeclarations>)</moduledeclarations>		
3	<moduledeclarations> <math>\rightarrow \epsilon</math></moduledeclarations>			
		<moduledeclarations>.syn_addr=NULL</moduledeclarations>		
		$free(\epsilon)$		
4	<moduledeclaration> → DECLARE MODULE ID SEMICOL</moduledeclaration>	<del>'</del>		
		<moduledeclaration>.addr=ID.addr</moduledeclaration>		
		free(DECLARE)		
		free(MODULE)		
		free(SEMICOL)		
5	<pre><othermodules(1)> → <module><othermodules(2)></othermodules(2)></module></othermodules(1)></pre>			
		<othermodules(1)>.addr=createNewNode(label:moduleNode,<module>.addr)</module></othermodules(1)>		
		<othermodules(1)>.syn_addr=addToStart(<othermodules(1)>.addr,<othermodules(2)>.syn_addr)</othermodules(2)></othermodules(1)></othermodules(1)>		
		free( <module>)</module>		
		free( <othermodules(2)></othermodules(2)>		
6	<othermodules> → ε</othermodules>			
		<othermodules>.syn_addr=NULL</othermodules>		
7	<pre><drivermodule> → DRIVERDEF DRIVER PROGRAM DRIVERENDDEF <moduledef></moduledef></drivermodule></pre>			
		<pre><drivermodule>.addr=createNewNode(label:driverModule,<moduledef>.addr)</moduledef></drivermodule></pre>		
		free(DRIVERDEF)		
		free(DRIVER)		
		free(PROGRAM)		
		free(DRIVERENDDEF)		
		free( <moduledef>)</moduledef>		
8	<module> → DEF MODULE ID ENDDEF TAKES INPUT SQBO <input_plist> SQBC SEMIC</input_plist></module>			
	· <del>-</del>	<module>.addr=createNewNode(label:module,ID.addr,<input_plist>.syn_addr,<ret>.addr,<moduledef>.addr)</moduledef></ret></input_plist></module>		
		free(DEF)		
		free(MODULE)		
		free(ENDDEF)		
		free(TAKES)		
		free(INPUT)		
		free(SQBO)		
		free( <input_plist>)</input_plist>		
		free(SQBC)		
		free(SEMICOL)		
	I .	free( <ret>)</ret>		

	free( <moduledef)< th=""></moduledef)<>
9 <ret> → RETURNS SQBO <output_plist> SQBC SEMICOL</output_plist></ret>	
- Act Arterorite again apar_phot again action	<ret>.addr=<output_plist>.syn_addr</output_plist></ret>
	free(RETURNS)
	free(SQBO)
	free( <output_plist>)</output_plist>
	free(SQBC)
	free(SEMICOL)
$0$ <ret> <math>\rightarrow \varepsilon</math></ret>	
	<ret>.addr=NULL</ret>
	$free(\varepsilon)$
1 <input_plist> → ID COLON <datatype> <input_plist2></input_plist2></datatype></input_plist>	
input_prior is a detail, por imput_prior	<input_plist>.addr=createNewNode(label:linkedListNode,ID.addr,<datatype>.addr)</datatype></input_plist>
	<pre>input_plist&gt;.syn_addr=addToStart(<input_plist>.addr,<input_plist2>.syn_addr)</input_plist2></input_plist></pre>
	free(ID)
	free(COLON)
+	free( <datatype>)</datatype>
+	free( <input_plist2>)</input_plist2>
2 <input_plist2(1)> → COMMA ID COLON <datatype> <input_plist2(2)></input_plist2(2)></datatype></input_plist2(1)>	
- Input_plistz(1)* / COMMINTED COLOR statially post simput_plistz(2)*	<input_plist2(1)>.addr=createNewNode(label:linkedListNode,ID.addr,<datatype>.addr)</datatype></input_plist2(1)>
	<pre>sinput_plist2(1)&gt;.syn_addr=addToStart(<input_plist2(1)>.addr,<input_plist2(2)>.syn_addr)</input_plist2(2)></input_plist2(1)></pre>
+	free(ID)
	free(COLON)
	free( <datatype>)</datatype>
+	free( <input_plist2>)</input_plist2>
	free(COMMA)
<b>3</b> <input_plist2> → ε</input_plist2>	TIEC(COMMIN)
- Input_plistz	<input_plist2>.syn_addr=NULL</input_plist2>
	free( $\varepsilon$ )
4 <output_plist> → ID COLON <type> <output_plist2></output_plist2></type></output_plist>	
- soutput_plists / ID OOLON stypes soutput_plist2s	<output_plist>.addr=createNewNode(label:linkedListNode,ID.addr,<type>.addr)</type></output_plist>
	<pre></pre> <pre><pre></pre> <pre></pre> &lt;</pre>
	free(ID)
	free(COLON)
	illee(collon)
<u> </u>	free( <type>)</type>
	free( <type>)  free(<outnut_plist2>)</outnut_plist2></type>
Southut nlist2(1)> > COMMA ID COLON < type> < outnut nlist2(2)>	free( <type>) free(<output_plist2>)</output_plist2></type>
5 <output_plist2(1)> → COMMA ID COLON <type> <output_plist2(2)></output_plist2(2)></type></output_plist2(1)>	free( <output_plist2>)</output_plist2>
5 <output_plist2(1)> → COMMA ID COLON <type> <output_plist2(2)></output_plist2(2)></type></output_plist2(1)>	free( <output_plist2>)  <pre><output_plist2(1)>.addr=createNewNode(label:linkedListNode,ID.addr,<type>.addr)</type></output_plist2(1)></pre></output_plist2>
5 <output_plist2(1)> → COMMA ID COLON <type> <output_plist2(2)></output_plist2(2)></type></output_plist2(1)>	free( <output_plist2>)  <pre></pre></output_plist2>
5 <output_plist2(1)> → COMMA ID COLON <type> <output_plist2(2)></output_plist2(2)></type></output_plist2(1)>	free( <output_plist2>)  <pre></pre></output_plist2>
5 <output_plist2(1)> → COMMA ID COLON <type> <output_plist2(2)></output_plist2(2)></type></output_plist2(1)>	free( <output_plist2>)  <pre></pre></output_plist2>
5 <output_plist2(1)> → COMMA ID COLON <type> <output_plist2(2)></output_plist2(2)></type></output_plist2(1)>	free( <output_plist2>)  <pre></pre></output_plist2>
5 <output_plist2(1)> → COMMA ID COLON <type> <output_plist2(2)></output_plist2(2)></type></output_plist2(1)>	free( <output_plist2>)  <pre></pre></output_plist2>
	free( <output_plist2>)  <pre></pre></output_plist2>
5 <output_plist2(1)> → COMMA ID COLON <type> <output_plist2(2)> 6 <output_plist2> → ε</output_plist2></output_plist2(2)></type></output_plist2(1)>	free( <output_plist2>)  <pre></pre></output_plist2>
	free( <output_plist2>)  <pre></pre></output_plist2>
6 <output_plist2> → ε</output_plist2>	free( <output_plist2>)  <pre></pre></output_plist2>
	free( <output_plist2>)  <pre></pre></output_plist2>

	<datatype>.addr = REAL</datatype>
19 <datatype> → BOOLEAN</datatype>	
	<datatype>.addr = BOOLEAN</datatype>
20 <datatype> → ARRAY SQBO <arr_range> SQBC OF <type></type></arr_range></datatype>	7 · · · · · · · · · · · · · · · · · · ·
	<datatype>.addr=createNewNode(label:arrayDataType,<arr_range>.addr,<type>.addr)</type></arr_range></datatype>
	free(ARRAY)
	free(SQBO)
	free( <arr_range>)</arr_range>
	free(SQBC)
	free(OF)
	free( <type>)</type>
21 <arr_range>→ <arr_index(1)>&lt; RANGEOP <arr_index(2)></arr_index(2)></arr_index(1)></arr_range>	
	<arr_range>.addr=createNewNode(label:rangeOP,<arr_index(1)>.addr,<arr_index(2)>.addr)</arr_index(2)></arr_index(1)></arr_range>
	free(RANGEOP)
	free( <arr_index(1)>)</arr_index(1)>
	free( <arr_index(2)>)</arr_index(2)>
22 <type> → INTEGER</type>	•
	<type>.addr = INTEGER</type>
23 <type> → REAL</type>	•
	<type>.addr = REAL</type>
24 <type> → BOOLEAN</type>	
	<type>.addr = BOOLEAN</type>
25 <moduledef> →START <statements> END</statements></moduledef>	
	<moduledef>.addr=<statements>.syn_addr</statements></moduledef>
	free(START)
	free(END)
	free( <statements>)</statements>
26 <statements(1)>→ <statement> <statements(2)></statements(2)></statement></statements(1)>	·
	<statements(1)>.syn_addr=addToStart(<statement>.addr,<statements(2)>.syn_addr)</statements(2)></statement></statements(1)>
	free( <statement>)</statement>
	free( <statements(2)>)</statements(2)>
27 <statements>→ ε</statements>	•
	<statements>.syn_addr=NULL</statements>
28 <statement> → <iostmt></iostmt></statement>	
	<statement>.addr=<iostmt>.addr</iostmt></statement>
29 <statement> → <simplestmt></simplestmt></statement>	
	<statement>.addr=<simplestmt>.addr</simplestmt></statement>
30 <statement> → <declarestmt></declarestmt></statement>	·
	<statement>.addr=<declarestmt>.addr</declarestmt></statement>
31 <statement> → <conditionalstmt></conditionalstmt></statement>	
	<statement>.addr=<conditionalstmt>.addr</conditionalstmt></statement>
32 <statement> → <iterativestmt></iterativestmt></statement>	
	<statement>.addr=<iterativestmt>.addr</iterativestmt></statement>
33 <iostmt> → PRINT BO <print_var> BC SEMICOL</print_var></iostmt>	
	<iostmt>.addr=createNewNode(label:printOutput,<print_var>.syn_addr)</print_var></iostmt>
	free(PRINT)
	free(BO)
	free(BC)
	free(SEMICOL)
34 <iostmt> → GET_VALUE BO ID BC SEMICOL</iostmt>	•

	25 Olycle and decreased Mr. Made (label and last UD and A
	<iostmt>.addr=createNewNode(label:getInput,ID.addr)</iostmt>
	free(GET_VALUE)
	free(BO)
	free(BC)
	free(SEMICOL)
35 <boolconst> →TRUE</boolconst>	
	<pre><boolconst>.addr=TRUE.addr</boolconst></pre>
36 <boolconst> → FALSE</boolconst>	
	<pre><boolconst>.addr=FALSE.addr</boolconst></pre>
<b>37</b> <print_var> → ID <n1></n1></print_var>	
	<print_var>.addr = ID.addr</print_var>
<n1>.inh_addr=<print_var>.addr</print_var></n1>	
	<print_var>.syn_addr=<n1>.syn_addr</n1></print_var>
	free( <n1>)</n1>
38 <print_var> → NUM</print_var>	
	<print_var>.syn_addr=NUM.addr</print_var>
39 <print_var> → RNUM</print_var>	
	<print_var>.syn_addr=RNUM.addr</print_var>
40 <print_var> → <boolconst></boolconst></print_var>	· - · -
	<print_var>.syn_addr=boolConst.addr</print_var>
41 <n1> → SQBO<sign><index2> SQBC</index2></sign></n1>	
	<n1>.addr=createNewNode(label:signedNum,<sign>.addr,<index2>.addr)</index2></sign></n1>
	<n1>.syn_addr=createNewNode(label:arrayElement,<n1>.inh_addr,<n1>.addr)</n1></n1></n1>
	free(SQBO)
	free(SQBC)
	free( <sign>)</sign>
40 (1)(4)	free( <index2>)</index2>
42 $\langle N1 \rangle \rightarrow \epsilon$	
	<n1>.syn_addr=<n1>.inh_addr</n1></n1>
43 <simplestmt> →<assignmentstmt></assignmentstmt></simplestmt>	
	<simplestmt>.addr=<assignmentstmt>.syn_addr</assignmentstmt></simplestmt>
	free( <assignmentstmt>)</assignmentstmt>
44 <simplestmt> → <modulereusestmt></modulereusestmt></simplestmt>	
	<simplestmt>.addr=<modulereusestmt>.syn_addr</modulereusestmt></simplestmt>
	free( <modulereusestmt>)</modulereusestmt>
45 <assignmentstmt> → ID <whichstmt></whichstmt></assignmentstmt>	
	<assignmentstmt>.addr=ID.addr</assignmentstmt>
<whichstmt>.inh_addr=<assignmentstmt>.addr</assignmentstmt></whichstmt>	
	<assignementstmt>.syn_addr=<whichstmt>.syn_addr</whichstmt></assignementstmt>
	free( <whichstmt>)</whichstmt>
46 <whichstmt> →<lvalueidstmt></lvalueidstmt></whichstmt>	•
<pre><!--valueIDStmt-->.inh_addr=<whichstmt>.inh_addr</whichstmt></pre>	
	<whichstmt>.syn_addr=<lvalueidstmt>,syn_addr</lvalueidstmt></whichstmt>
	free( <lvalueidstmt>)</lvalueidstmt>
47 <whichstmt> → <lvaluearrstmt></lvaluearrstmt></whichstmt>	
<pre></pre> <pre>&lt;</pre>	
Transport Courte and Laudi - Swillonounte and Laudi	<pre><whichstmt>.syn_addr=<lvaluearrstmt>,syn_addr</lvaluearrstmt></whichstmt></pre>
40 displica IDC4m45 ACCIONIOD designations OFMICOL	free( <ivaluearrstmt>)</ivaluearrstmt>
48 <lvalueidstmt> → ASSIGNOP <expression> SEMICOL</expression></lvalueidstmt>	diet albeitet er eine Arte annie Albeitet in die 1800 in die 1
	<pre><lvalueidstmt>.syn_addr=createNewNode(label:lvalueIDStmt,<lvalueidstmt>.inh_addr,<expression>.syn_addr)</expression></lvalueidstmt></lvalueidstmt></pre>

	free(ASSIGNOP)
	free( <expression>)</expression>
	free(SEMICOL)
49 <lvaluearrstmt> → SQBO <exprindex> SQBC ASSIGNOP <expression> SEMICOL</expression></exprindex></lvaluearrstmt>	
	<pre><!--valueARRStmt-->.addr=createNewNode(label:arrayElement,<!--valueARRStmt-->.inh_addr,<exprindex>.syn_addr)</exprindex></pre>
	<pre><lvaluearrstmt>.syn_addr=createNewNode(label:lvalueARRStmt,<lvaluearrstmt>.addr,<expression>.syn_index)</expression></lvaluearrstmt></lvaluearrstmt></pre>
	free(SQBO)
	free(SQBC)
	free(exprIndex)
	free(ASSIGNOP)
	free( <expression>)</expression>
	free(SEMICOL)
50 <arr_index> → <sign><index2></index2></sign></arr_index>	
	<arr_index>.addr=createNewNode(label:signedNum,<sign>.addr,<index2>.addr)</index2></sign></arr_index>
	free( <sign>)</sign>
	free( <index2>)</index2>
51 <index2> → ID</index2>	•
	<index2>.addr=ID.addr</index2>
52 <index2> → NUM</index2>	
	<index2>.addr=NUM.addr</index2>
53 $\langle \text{sign} \rangle \rightarrow \epsilon$	
sign of the same o	<sign>.addr=NULL</sign>
54 <sign> → MINUS</sign>	ogn add trott
Sign Vinites	<sign>.addr=MINUS.addr</sign>
55 <sign> → PLUS</sign>	Sign addi-ininocaddi
Signe 71 200	<sign>.addr=PLUS.addr</sign>
56 <modulereusestmt> → <optional> USE MODULE ID WITH PARAMETERS <actual_para_list> SEMICOL</actual_para_list></optional></modulereusestmt>	Signification   Eco.addi
**************************************	<pre><modulereusestmt>.addr=createNewNode(label:moduleReuseStmt,ID.addr,<actual_para_list>.syn_addr)</actual_para_list></modulereusestmt></pre>
<pre><pre><pre><optional>.inh_addr=<modulereusestmt>.addr</modulereusestmt></optional></pre></pre></pre>	**************************************
sophorials init_addr = should reascounts laddr	<pre><modulereusestmt>.syn_addr=<optional>.syn_addr</optional></modulereusestmt></pre>
	free( <optional>)</optional>
	free(USE)
	free(MODULE)
	free(WITH)
	free(PARAMETERS)
	free( <actual_para_list>)</actual_para_list>
E7 costual nara lists a saigns also ships	free(SEMICOL)
57 <actual_para_list> → <sign> <k> <n9></n9></k></sign></actual_para_list>	
	<pre><actual_para_list>.addr=createNewNode(label:signedValue,<sign>.addr,<k>.syn_addr)</k></sign></actual_para_list></pre>
	<pre><actual_para_list>.syn_addr=addToStart(<actual_para_list>.addr,<n9>.syn_addr)</n9></actual_para_list></actual_para_list></pre>
	free( <sign>)</sign>
	free( <k>)</k>
	free( <n9>)</n9>
58 <n9(1)> → COMMA <sign> <k> <n9(2)></n9(2)></k></sign></n9(1)>	T.,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
	<pre><n9(1)>.addr=createNewNode(label:signedValue,<sign>.addr,<k>.syn_addr)</k></sign></n9(1)></pre>
	<pre><n9(1)>.syn_addr=addToStart(<n9(1)>.addr,<n9(2)>.syn_addr)</n9(2)></n9(1)></n9(1)></pre>
	free(COMMA)
	free( <sign>)</sign>
	free( <k>) free(<n9(2)>)</n9(2)></k>

59	$<$ N9> $\rightarrow \epsilon$	
		<n9>.syn_addr=NULL</n9>
60	$\langle K \rangle \rightarrow NUM$	
		<k>.syn_addr = NUM.addr</k>
61	$\langle K \rangle \rightarrow RNUM$	
		<k>.syn_addr = RNUM.addr</k>
62	$<$ K $> \rightarrow <$ boolConst $>$	
		<k>.syn_addr=<boolconst>.addr</boolconst></k>
		free( <boolconst>)</boolconst>
63	<k> → ID <actual_para_list2></actual_para_list2></k>	
		<k>.addr=ID.addr</k>
	<actual_para_list2>.inh_addr=<k>.addr</k></actual_para_list2>	
		<k>.syn_addr=<actual_para_list2>.syn_addr</actual_para_list2></k>
		free( <actual_para_list2>)</actual_para_list2>
64	<actual_para_list2> → SQBO <exprindex> SQBC</exprindex></actual_para_list2>	
	<b>-</b>	<actual_para_list2>.syn_addr=<exprindex>.syn_addr</exprindex></actual_para_list2>
		free(SQBO)
		free( <exprindex>)</exprindex>
		free(SQBC)
65	<actual_para_list2> → ε</actual_para_list2>	
		<actual_para_list2>.syn_addr=<actual_para_list>.inh_addr</actual_para_list></actual_para_list2>
		free(ε)
66	<pre><optional> → SQBO <idlist> SQBC ASSIGNOP</idlist></optional></pre>	
- 00	Topinoliais 7 OQDO TINEISIS OQDO NOCIONOL	<pre><optional>.syn_addr=createNewNode(label:assignOp,<idlist>.syn_addr,<optional>.inh_addr)</optional></idlist></optional></pre>
		free(SQBO)
		free(sidList>)
		free(SQBC)
		free(ASSIGNOP)
67	continuels	III EE(ASSIGNOP)
67	<pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre>	continuals are addressed into address
		<pre><optional>.syn_addr=<optional>.inh_addr</optional></optional></pre>
00		$free(\epsilon)$
68	<idlist> -&gt; ID <idlist2></idlist2></idlist>	
		<idlist>.syn_addr=addToStart(ID.addr,<idlist2>.syn_addr)</idlist2></idlist>
		free( <idlist2>)</idlist2>
69	<idlist2(1)> -&gt; COMMA ID <idlist2(2)></idlist2(2)></idlist2(1)>	
		<idlist2(1)>.syn_addr=addToStart(ID.addr,<idlist2(2)>.syn_addr)</idlist2(2)></idlist2(1)>
		free(COMMA)
		free( <idlist2(2)>)</idlist2(2)>
70	<idlist2> -&gt; ε</idlist2>	
		<idlist2>.syn_addr=NULL</idlist2>
		$free(\epsilon)$
71	<expression> -&gt; <abexpr></abexpr></expression>	
		<expression>.syn_addr=<abexpr>.syn_addr</abexpr></expression>
		free( <abexpr>)</abexpr>
72	<expression> -&gt; <u></u></expression>	
		<expression>.syn_addr=<u>.syn_addr</u></expression>
		free( <u>)</u>
73	<u> -&gt; <unary_op> <non_term></non_term></unary_op></u>	
		<u>.syn_addr=createNewNode(label:unaryOp,<unaryop>.addr,<non_term>.syn_addr)</non_term></unaryop></u>
		free( <unary_op>)</unary_op>

	free( <non_term>)</non_term>
74 <non_term> -&gt; BO <arithmeticexpr> BC</arithmeticexpr></non_term>	
	<non_term>.syn_addr=<arithmeticexpr>.syn_addr</arithmeticexpr></non_term>
	free(BO)
	free( <arithmeticexpr>)</arithmeticexpr>
	free(BC)
75 <non_term> -&gt; <var_const></var_const></non_term>	
	<non_term>.syn_addr=<var_const>.addr</var_const></non_term>
	free( <var_const>)</var_const>
<b>76</b> <unary_op> → PLUS</unary_op>	
	<unary_op>.addr = PLUS</unary_op>
77 <unary_op> → MINUS</unary_op>	
	<unary_op>.addr = MINUS</unary_op>
<b>78</b> <abexpr> → <anyterm> <n7></n7></anyterm></abexpr>	
	<abexpr>.addr=<anyterm>.syn_addr</anyterm></abexpr>
<n7>.inh_addr=<abexpr>.addr</abexpr></n7>	
	<abexpr>.syn_addr=<n7>.syn_addr</n7></abexpr>
	free(AnyTerm)
	free( <n7>)</n7>
<b>79</b> <n7(1)> → <logicalop><anyterm><n7(2)></n7(2)></anyterm></logicalop></n7(1)>	•
	<n7(1)>.addr=createNewNode(label:<logicalop>,<n7(1)>.inh_addr,<anyterm>.syn_addr)</anyterm></n7(1)></logicalop></n7(1)>
<n7(2)>.inh_addr=<n7(1)>.addr</n7(1)></n7(2)>	
	<n7(1)>.syn_addr=<n7(2)>.syn_addr</n7(2)></n7(1)>
	free( <logicalop>)</logicalop>
	free( <anyterm>)</anyterm>
	free( <n7(2)></n7(2)>
<b>80</b> <n7> → ε</n7>	
	<n7>.syn_addr=<n7>.inh_addr</n7></n7>
	$free(\epsilon)$
81 <anyterm> → <arithmeticexpr> <n8></n8></arithmeticexpr></anyterm>	
	<anyterm>.addr=<arithmeticexpr>.syn_addr</arithmeticexpr></anyterm>
<n8>.inh_addr=<anyterm>.addr</anyterm></n8>	
	<anyterm>.syn_addr=<n8>.syn_addr</n8></anyterm>
	free( <arithmeticexpr>)</arithmeticexpr>
	free( <n8>)</n8>
82 <anyterm> → <boolconst></boolconst></anyterm>	
	<anyterm>.syn_addr=<boolconst>.addr</boolconst></anyterm>
	free( <boolconst>)</boolconst>
83 <n8> → <relationalop> <arithmeticexpr></arithmeticexpr></relationalop></n8>	
83 <n8> → <relationalop> <arithmeticexpr></arithmeticexpr></relationalop></n8>	<pre></pre> <pre>&lt;</pre>
83 <n8> → <relationalop> <arithmeticexpr></arithmeticexpr></relationalop></n8>	
83 <n8> → <relationalop> <arithmeticexpr></arithmeticexpr></relationalop></n8>	<pre></pre> <pre>&lt;</pre>
<ul> <li>83 <n8> → <relationalop> <arithmeticexpr></arithmeticexpr></relationalop></n8></li> <li>84 <n8> → ε</n8></li> </ul>	<pre></pre> <pre><n8>.syn_addr=createNewNode(label:<relationalop>,<n8>.inh_addr,<arithmeticexpr>.syn_addr) free(<relationalop>)</relationalop></arithmeticexpr></n8></relationalop></n8></pre>
	<pre></pre> <pre><n8>.syn_addr=createNewNode(label:<relationalop>,<n8>.inh_addr,<arithmeticexpr>.syn_addr) free(<relationalop>)</relationalop></arithmeticexpr></n8></relationalop></n8></pre>
	<pre></pre> <pre>// CarithmeticExpr&gt;)</pre> <pre> <pre></pre> <pre> <pre></pre> <pre></pre> <pre></pre> <pre></pre> <pre></pre> <pre></pre> <pre>// CarithmeticExpr&gt;)</pre></pre></pre>
	<pre></pre> <pre>&lt;</pre>
84 <n8> → ε</n8>	<pre></pre> <pre>&lt;</pre>
84 <n8> → ε</n8>	<pre></pre> <pre><n8>.syn_addr=createNewNode(label:<relationalop>,<n8>.inh_addr,<arithmeticexpr>.syn_addr) free(<relationalop>) free(<arithmeticexpr>)  <pre><n8>.syn_addr=<n8>.inh_addr free(ε)</n8></n8></pre></arithmeticexpr></relationalop></arithmeticexpr></n8></relationalop></n8></pre>
84 <n8> → ε  85 <var_const> → ID</var_const></n8>	<pre></pre> <pre><n8>.syn_addr=createNewNode(label:<relationalop>,<n8>.inh_addr,<arithmeticexpr>.syn_addr) free(<relationalop>) free(<arithmeticexpr>)  <pre><n8>.syn_addr=<n8>.inh_addr free(ε)</n8></n8></pre></arithmeticexpr></relationalop></arithmeticexpr></n8></relationalop></n8></pre>

	<var_const>.addr = RNUM</var_const>
88 <arithmeticexpr> → <term> <arithmeticexpr2></arithmeticexpr2></term></arithmeticexpr>	
	<arithmeticexpr>.addr=<term>.syn_addr</term></arithmeticexpr>
<arithmeticexpr2>.inh_addr=<arithmeticexpr>.addr</arithmeticexpr></arithmeticexpr2>	
	<arithmeticexpr>.syn_addr=<arithmeticexpr2>.syn_addr</arithmeticexpr2></arithmeticexpr>
	free( <term>)</term>
	free( <arithmeticexpr2>)</arithmeticexpr2>
89 <arithmeticexpr2(1)> → <low_op> <term> <arithmeticexpr2(2)></arithmeticexpr2(2)></term></low_op></arithmeticexpr2(1)>	
	<arithmeticexpr2(1)>.addr=createNewNode(label:<low_op>,<arithmeticexpr2(1)>inh_addr,<term>.syn_addr)</term></arithmeticexpr2(1)></low_op></arithmeticexpr2(1)>
<arithmeticexpr2(2)>.inh_addr=<arithmeticexpr2(1)>.addr</arithmeticexpr2(1)></arithmeticexpr2(2)>	
	<arithmeticexpr2(1)>.syn_addr=<arithmeticexpr2(2)>.syn_addr</arithmeticexpr2(2)></arithmeticexpr2(1)>
	free( <low_op>)</low_op>
	free( <term>)</term>
	free( <arithmeticexpr2(2)>)</arithmeticexpr2(2)>
90 <arithmeticexpr2> <math>\rightarrow \epsilon</math></arithmeticexpr2>	
	<arithmeticexpr2>.syn_addr=<arithmeticexpr2>.inh_addr</arithmeticexpr2></arithmeticexpr2>
91 <term> → <factor> <term2></term2></factor></term>	
	<term>.addr=<factor>.syn_addr</factor></term>
<term2>.inh_addr=<term>.addr</term></term2>	
	<term>.syn_addr=<term2>.syn_addr</term2></term>
	free( <factor>)</factor>
	free( <term2>)</term2>
92 <term2> <math>\rightarrow \epsilon</math></term2>	
	<term2>.syn_addr=<term2>.inh_addr</term2></term2>
93 $<$ term2(1)> $\rightarrow$ $<$ high_op> $<$ factor> $<$ term2(2)>	
	<term2(1)>.addr=createNewNode(label:<high_op>,<term2(1)>.inh_addr,<factor>.syn_addr)</factor></term2(1)></high_op></term2(1)>
<term2(2)>.inh_addr=<term2(1)>.addr</term2(1)></term2(2)>	
	<term2(2)>.syn_addr=<term2(1)>.syn_addr</term2(1)></term2(2)>
	free( <high_op>)</high_op>
	free( <factor>)</factor>
	free( <term2(2)>)</term2(2)>
94 <factor> → BO <abexpr> BC</abexpr></factor>	
	<pre><factor>.syn_addr=<abexpr>.syn_addr</abexpr></factor></pre>
	free(BO)
	free( <abexpr>)</abexpr>
	free(BC)
95 <factor> -&gt; NUM</factor>	
	<factor>.syn_addr=NUM.addr</factor>
96 <factor> -&gt; RNUM</factor>	
	<factor>.syn_addr=RNUM.addr</factor>
97 <factor> -&gt; <boolconst></boolconst></factor>	
	<factor>.syn_addr=<boolconst>.addr</boolconst></factor>
	free( <boolconst>)</boolconst>
98 <factor> -&gt; ID <factor2></factor2></factor>	
	<factor>.addr=ID.addr</factor>
<factor2>.inh_addr=<factor>.addr</factor></factor2>	
	<pre><factor>.syn_addr=<factor2>.syn_addr</factor2></factor></pre>
	<pre><factor>.syn_addr=<factor2>.syn_addr free(ID)</factor2></factor></pre>
	<pre><factor>.syn_addr=<factor2>.syn_addr</factor2></factor></pre>

	<factor2>.syn_addr=<factor2>.inh_addr</factor2></factor2>
00 <factor2> -&gt; SQBO <exprindex> SQBC</exprindex></factor2>	
	<factor2>.syn_addr=createNewNode(label:arrayExprIndex,<factor2>.inh_addr,<exprindex>.syn_addr)</exprindex></factor2></factor2>
	free(SQBO)
	free(SQBC)
	free( <exprindex>)</exprindex>
01 <exprindex> → <sign> <exprindex2></exprindex2></sign></exprindex>	
	<exprindex>.syn_addr=createNewNode(label:signedExprIndex,<sign>.addr,<exprindex2>.syn_addr)</exprindex2></sign></exprindex>
	free( <sign>)</sign>
	free( <exprindex2>)</exprindex2>
02 <exprindex> → <arrexpr></arrexpr></exprindex>	
	<exprindex>.syn_addr=<arrexpr>.syn_addr</arrexpr></exprindex>
	free( <arrexpr>)</arrexpr>
03 <exprindex2> → <index2></index2></exprindex2>	
	<exprindex2>.syn_addr=<index2>.syn_addr</index2></exprindex2>
	free( <index2>)</index2>
04 <exprindex2> → BO <arrexpr> BC</arrexpr></exprindex2>	
	<exprindex2>.syn_addr=<arrexpr>.syn_addr</arrexpr></exprindex2>
	free(BO)
	free( <arrexpr>)</arrexpr>
	free(BC)
05 <arrexpr> → <arrterm> <arrexpr2></arrexpr2></arrterm></arrexpr>	
	<arrexpr>.addr=<arrterm>.syn_addr</arrterm></arrexpr>
<arrexpr2>.inh_addr=<arrexpr>.addr</arrexpr></arrexpr2>	
	<arrexpr>.syn_addr=<arrexpr2>.syn_addr</arrexpr2></arrexpr>
	free( <arrterm>)</arrterm>
	free( <arrexpr2>)</arrexpr2>
<b>06</b> <arrexpr2> → ε</arrexpr2>	
	<arrexpr2>.syn_addr=<arrexpr2>.inh_addr</arrexpr2></arrexpr2>
07 <arrexpr2(1)> → <low_op> <arrterm> <arrexpr2(2)></arrexpr2(2)></arrterm></low_op></arrexpr2(1)>	
	<arrexpr2(1)>.addr=createNewNode(label:<low_op>,<arrexpr2(1)>.inh_addr,<arrterm>.syn_addr)</arrterm></arrexpr2(1)></low_op></arrexpr2(1)>
<arrexpr2(2)>.inh_addr=<arrexpr2(1)>.addr</arrexpr2(1)></arrexpr2(2)>	
	<arrexpr2(1)>.syn_addr=<arrexpr2(2)>.syn_addr</arrexpr2(2)></arrexpr2(1)>
	<arrexpr2(1)>.syn_addr=<arrexpr2(2)>.syn_addr free(<low_op>)</low_op></arrexpr2(2)></arrexpr2(1)>
	free( <low_op>)</low_op>
08 <arrterm> → <arrfactor> <arrterm2></arrterm2></arrfactor></arrterm>	free( <low_op>) free(<arrterm>)</arrterm></low_op>
18 <arrterm> → <arrterm2></arrterm2></arrterm>	free( <low_op>) free(<arrterm>)</arrterm></low_op>
08 <arrterm> → <arrfactor> <arrterm2> <arrterm2>.inh_addr=<arrterm>.addr</arrterm></arrterm2></arrterm2></arrfactor></arrterm>	free( <low_op>) free(<arrterm>) free(<arrexpr2(2)>)</arrexpr2(2)></arrterm></low_op>
	free( <low_op>) free(<arrterm>) free(<arrexpr2(2)>)</arrexpr2(2)></arrterm></low_op>
	free( <low_op>) free(<arrterm>) free(<arrexpr2(2)>)  <arrterm>.addr=<arrfactor>.syn_addr</arrfactor></arrterm></arrexpr2(2)></arrterm></low_op>
	free( <low_op>) free(<arrterm>) free(<arrexpr2(2)>)  <arrterm>.addr=<arrfactor>.syn_addr <arrterm>.syn_addr=<arrterm2>.syn_addr</arrterm2></arrterm></arrfactor></arrterm></arrexpr2(2)></arrterm></low_op>
	free( <low_op>) free(<arrterm>) free(<arrexpr2(2)>)  <arrterm>.addr=<arrfactor>.syn_addr <arrterm>.syn_addr=<arrterm2>.syn_addr free(<arrfactor>)</arrfactor></arrterm2></arrterm></arrfactor></arrterm></arrexpr2(2)></arrterm></low_op>
<arrterm2>.inh_addr=<arrterm>.addr</arrterm></arrterm2>	free( <low_op>) free(<arrterm>) free(<arrexpr2(2)>)  <arrterm>.addr=<arrfactor>.syn_addr <arrterm>.syn_addr=<arrterm2>.syn_addr free(<arrfactor>)</arrfactor></arrterm2></arrterm></arrfactor></arrterm></arrexpr2(2)></arrterm></low_op>
<arrterm2>.inh_addr=<arrterm>.addr</arrterm></arrterm2>	free( <low_op>) free(<arrterm>) free(<arrexpr2(2)>)  <arrterm>.addr=<arrfactor>.syn_addr <arrterm>.syn_addr=<arrterm2>.syn_addr free(<arrfactor>) free(<arrterm2>)</arrterm2></arrfactor></arrterm2></arrterm></arrfactor></arrterm></arrexpr2(2)></arrterm></low_op>
<pre><arrterm2>.inh_addr=<arrterm>.addr  09 <arrterm2(1)> → <high_op> <arrfactor> <arrterm2(2)></arrterm2(2)></arrfactor></high_op></arrterm2(1)></arrterm></arrterm2></pre>	free( <low_op>) free(<arrterm>) free(<arrexpr2(2)>)  <arrterm>.addr=<arrfactor>.syn_addr <arrterm>.syn_addr=<arrterm2>.syn_addr free(<arrfactor>) free(<arrterm2>)</arrterm2></arrfactor></arrterm2></arrterm></arrfactor></arrterm></arrexpr2(2)></arrterm></low_op>
<pre><arrterm2>.inh_addr=<arrterm>.addr  09 <arrterm2(1)> → <high_op> <arrfactor> <arrterm2(2)></arrterm2(2)></arrfactor></high_op></arrterm2(1)></arrterm></arrterm2></pre>	free( <low_op>) free(<arrterm>) free(<arrexpr2(2)>)  <arrterm>.addr=<arrfactor>.syn_addr  <arrterm>.syn_addr=<arrterm2>.syn_addr  free(<arrfactor>) free(<arrterm2>)  <arrterm2(1)>.addr=createNewNode(label:<high_op>,<arrterm2(1)>.inh_addr,<arrfactor>.syn_addr)</arrfactor></arrterm2(1)></high_op></arrterm2(1)></arrterm2></arrfactor></arrterm2></arrterm></arrfactor></arrterm></arrexpr2(2)></arrterm></low_op>
<pre><arrterm2>.inh_addr=<arrterm>.addr  09 <arrterm2(1)> → <high_op> <arrfactor> <arrterm2(2)></arrterm2(2)></arrfactor></high_op></arrterm2(1)></arrterm></arrterm2></pre>	free( <arrterm>) free(<arrexpr2(2)>)  <arrterm>.addr=<arrterm2.syn_addr <arrterm="">.syn_addr=<arrterm2>.syn_addr  free(<arrterm2>)  free(<arrterm2>)  <arrterm2(1)>.addr=createNewNode(label:<high_op>,<arrterm2(1)>.inh_addr,<arrfactor>.syn_addr)  <arrterm2(1)>.syn_addr=<arrterm2(2)>,syn_addr <arrterm2(1)>.syn_addr=<arrterm2(2)>,syn_addr <arrterm2(1)>.syn_addr=<arrterm2(2)>,syn_addr <arrterm2(1)>.syn_addr=<arrterm2(2)>,syn_addr <arrterm2(1)>.syn_addr=<arrterm2(2)>,syn_addr <arrterm2(1)>.syn_addr=<arrterm2(2)>,syn_addr <arrterm2(1)>.syn_addr=<arrterm2(2)>,syn_addr <arrterm2(1)>.syn_addr=<arrterm2(2)>,syn_addr <arrterm2(1)>.syn_addr=<arrterm2(2)>,syn_addr <arrterm2(1)>.syn_addr=<arrterm2(2)>,syn_addr</arrterm2(2)></arrterm2(1)></arrterm2(2)></arrterm2(1)></arrterm2(2)></arrterm2(1)></arrterm2(2)></arrterm2(1)></arrterm2(2)></arrterm2(1)></arrterm2(2)></arrterm2(1)></arrterm2(2)></arrterm2(1)></arrterm2(2)></arrterm2(1)></arrterm2(2)></arrterm2(1)></arrterm2(2)></arrterm2(1)></arrfactor></arrterm2(1)></high_op></arrterm2(1)></arrterm2></arrterm2></arrterm2></arrterm2.syn_addr></arrterm></arrexpr2(2)></arrterm>

	<arrterm2>.syn_addr=<arrterm2>.inh_addr</arrterm2></arrterm2>
111 <arrfactor> → ID</arrfactor>	-an termize sayin_addi = and termize simin_addi
am actor — ib	<arrfactor>.syn_addr=ID.addr</arrfactor>
112 <arrfactor> → NUM</arrfactor>	-am actors.syn_addi-ib.addi
112 Valli actor > Note	<arrfactor>.syn_addr=NUM.addr</arrfactor>
113 <arrfactor> → <boolconst></boolconst></arrfactor>	Carri actors.syri_addr=Now.addr
TIS Valif actor → Spooleonst	<arrfactor>.syn_addr=<boolconst>.addr</boolconst></arrfactor>
	free( <boolconst>)</boolconst>
114 <arrfactor> → BO <arrexpr> BC</arrexpr></arrfactor>	mee(Nooloomstr)
THE CANT ACTOR A BO CANTEAUTY BO	<arrfactor>.syn_addr=<arrexpr>.syn_addr</arrexpr></arrfactor>
	free(BO)
	free(BC)
	free( <arrexpr>)</arrexpr>
115 <low_op> → PLUS</low_op>	inee(\angle angle xpi>)
115 NOW_OP > PLOS	<li><low_op>.addr = PLUS.addr</low_op></li>
116 clay on MINUS	Now_op>.addi = PLOS.addi
116 <low_op> → MINUS</low_op>	clays and address MINUIC address
117 chigh on MIII	<li><low_op>.addr = MINUS.addr</low_op></li>
117 <high_op> → MUL</high_op>	chieb on odde = MIII odde
440 Mills on DIV	<high_op>.addr = MUL.addr</high_op>
118 <high_op> → DIV</high_op>	Attaches and All Division Division
40 + 10 - 00	<high_op>.addr = DIV.addr</high_op>
119 <logicalop>→ OR</logicalop>	ductorious addition OR addition
400 de Carlo de AND	<li><logicalop>.addr = OR.addr</logicalop></li>
120 <logicalop>→ AND</logicalop>	duction of the AND of the
121 <relationalop>→ LT</relationalop>	<li><logicalop>.addr = AND.addr</logicalop></li>
121 <leiationalop>→ L1</leiationalop>	<relationalop>.addr = LT.addr</relationalop>
	PelationalOp>.addi = L1.addi
122 gradationalOne L. I. E.	
122 <relationalop>→ LE</relationalop>	crolationalOn> adds = LE adds
	<relationalop>.addr = LE.addr</relationalop>
122 <relationalop>→ LE  123 <relationalop>→ GT</relationalop></relationalop>	
123 <relationalop>→ GT</relationalop>	<pre><relationalop>.addr = LE.addr </relationalop></pre> <pre><relationalop>.addr = GT.addr</relationalop></pre>
	<relationalop>.addr = GT.addr</relationalop>
123 <relationalop>→ GT  124 <relationalop>→ GE</relationalop></relationalop>	
123 <relationalop>→ GT</relationalop>	<pre><relationalop>.addr = GT.addr </relationalop></pre> <pre><relationalop>.addr = GE.addr</relationalop></pre>
123 <relationalop>→ GT  124 <relationalop>→ GE  125 <relationalop>→ EQ</relationalop></relationalop></relationalop>	<relationalop>.addr = GT.addr</relationalop>
123 <relationalop>→ GT  124 <relationalop>→ GE</relationalop></relationalop>	<pre><relationalop>.addr = GT.addr  <relationalop>.addr = GE.addr  <relationalop>.addr = EQ.addr</relationalop></relationalop></relationalop></pre>
123 <relationalop>→ GT  124 <relationalop>→ GE  125 <relationalop>→ EQ  126 <relationalop>→ NE</relationalop></relationalop></relationalop></relationalop>	<pre><relationalop>.addr = GT.addr </relationalop></pre> <pre><relationalop>.addr = GE.addr</relationalop></pre>
123 <relationalop>→ GT  124 <relationalop>→ GE  125 <relationalop>→ EQ</relationalop></relationalop></relationalop>	<pre><relationalop>.addr = GT.addr  <relationalop>.addr = GE.addr  <relationalop>.addr = EQ.addr  <relationalop>.addr = NE.addr</relationalop></relationalop></relationalop></relationalop></pre>
123 <relationalop>→ GT  124 <relationalop>→ GE  125 <relationalop>→ EQ  126 <relationalop>→ NE</relationalop></relationalop></relationalop></relationalop>	<pre><relationalop>.addr = GT.addr  <relationalop>.addr = GE.addr  <relationalop>.addr = EQ.addr  <relationalop>.addr = NE.addr  <relationalop>.addr = NE.addr  </relationalop></relationalop></relationalop></relationalop></relationalop></pre> <pre> <pre></pre> &lt;</pre>
123 <relationalop>→ GT  124 <relationalop>→ GE  125 <relationalop>→ EQ  126 <relationalop>→ NE</relationalop></relationalop></relationalop></relationalop>	<pre></pre> <pre></pre> <pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre>
123 <relationalop>→ GT  124 <relationalop>→ GE  125 <relationalop>→ EQ  126 <relationalop>→ NE</relationalop></relationalop></relationalop></relationalop>	<pre><relationalop>.addr = GT.addr  <relationalop>.addr = GE.addr  <relationalop>.addr = EQ.addr  <relationalop>.addr = NE.addr  <relationalop>.addr = NE.addr  <declarestmt>.addr=createNewNode(label:declareStmt,<idlist>.syn_addr,<datatype>.addr)  free(DECLARE) free(<idlist>)</idlist></datatype></idlist></declarestmt></relationalop></relationalop></relationalop></relationalop></relationalop></pre>
123 <relationalop>→ GT  124 <relationalop>→ GE  125 <relationalop>→ EQ  126 <relationalop>→ NE</relationalop></relationalop></relationalop></relationalop>	<pre><relationalop>.addr = GT.addr  <relationalop>.addr = GE.addr  <relationalop>.addr = EQ.addr  <relationalop>.addr = NE.addr  </relationalop></relationalop></relationalop></relationalop></pre> <pre> </pre> <pre> <pre></pre> <pre></pre> <pre> <pre> <pre></pre> <pre> <pre></pre> <pre> <pre></pre> <pre> <pre>free(DECLARE)</pre> <pre> free(COLON)</pre></pre></pre></pre></pre></pre></pre>
123 <relationalop>→ GT  124 <relationalop>→ GE  125 <relationalop>→ EQ  126 <relationalop>→ NE</relationalop></relationalop></relationalop></relationalop>	<pre></pre> <pre></pre> <pre></pre> <pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre>
123 <relationalop>→ GT  124 <relationalop>→ GE  125 <relationalop>→ EQ  126 <relationalop>→ NE  127 <declarestmt>→ DECLARE <idlist> COLON <datatype> SEMICOL</datatype></idlist></declarestmt></relationalop></relationalop></relationalop></relationalop>	<pre><relationalop>.addr = GT.addr  <relationalop>.addr = GE.addr  <relationalop>.addr = EQ.addr  <relationalop>.addr = NE.addr  </relationalop></relationalop></relationalop></relationalop></pre> <pre> </pre> <pre> <pre></pre> <pre></pre> <pre> <pre> <pre></pre> <pre> <pre></pre> <pre> <pre></pre> <pre> <pre>free(DECLARE)</pre> <pre> free(COLON)</pre></pre></pre></pre></pre></pre></pre>
123 <relationalop>→ GT  124 <relationalop>→ GE  125 <relationalop>→ EQ  126 <relationalop>→ NE</relationalop></relationalop></relationalop></relationalop>	<pre></pre> <pre></pre> <pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre>
123 <relationalop>→ GT  124 <relationalop>→ GE  125 <relationalop>→ EQ  126 <relationalop>→ NE  127 <declarestmt>→ DECLARE <idlist> COLON <datatype> SEMICOL</datatype></idlist></declarestmt></relationalop></relationalop></relationalop></relationalop>	<pre><relationalop>.addr = GT.addr  <relationalop>.addr = GE.addr  <relationalop>.addr = EQ.addr  <relationalop>.addr = NE.addr  <relationalop>.addr = NE.addr  <declarestmt>.addr=createNewNode(label:declareStmt,<idlist>.syn_addr,<datatype>.addr)  free(DECLARE)  free(<idlist>)  free(<idlist>)  free(<olon) free(<datatype="">)  free(<datatype>)  free(SEMICOL)  </datatype></olon)></idlist></idlist></datatype></idlist></declarestmt></relationalop></relationalop></relationalop></relationalop></relationalop></pre> <pre> </pre> <pre>  <pre> </pre> <pre> <pre> <pre> </pre> <pre> <p< td=""></p<></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre>
123 <relationalop>→ GT  124 <relationalop>→ GE  125 <relationalop>→ EQ  126 <relationalop>→ NE  127 <declarestmt>→ DECLARE <idlist> COLON <datatype> SEMICOL</datatype></idlist></declarestmt></relationalop></relationalop></relationalop></relationalop>	<pre><relationalop>.addr = GT.addr  <relationalop>.addr = GE.addr  <relationalop>.addr = EQ.addr  <relationalop>.addr = NE.addr  <relationalop>.addr = NE.addr  </relationalop></relationalop></relationalop></relationalop></relationalop></pre> <pre> <pre></pre></pre>
123 <relationalop>→ GT  124 <relationalop>→ GE  125 <relationalop>→ EQ  126 <relationalop>→ NE  127 <declarestmt>→ DECLARE <idlist> COLON <datatype> SEMICOL</datatype></idlist></declarestmt></relationalop></relationalop></relationalop></relationalop>	<pre></pre> <pre>&lt;</pre>
123 <relationalop>→ GT  124 <relationalop>→ GE  125 <relationalop>→ EQ  126 <relationalop>→ NE  127 <declarestmt>→ DECLARE <idlist> COLON <datatype> SEMICOL</datatype></idlist></declarestmt></relationalop></relationalop></relationalop></relationalop>	<pre><relationalop>.addr = GT.addr  <relationalop>.addr = GE.addr  <relationalop>.addr = EQ.addr  <relationalop>.addr = NE.addr  <relationalop>.addr = NE.addr  </relationalop></relationalop></relationalop></relationalop></relationalop></pre> <pre> <pre></pre></pre>

	free( <casestmts>)</casestmts>
	free( <defaultstmts>)</defaultstmts>
	free(END)
129 <casestmts>→ CASE <value> COLON <statements> BREAK SEMICOL <casestmts2< td=""><td></td></casestmts2<></statements></value></casestmts>	
	<casestmts>.addr=createNewNode(label:caseValueStmts,<value>.addr,<statements>.syn_addr)</statements></value></casestmts>
	<casestmts>.syn_addr=addToStart(<casestmts>.addr,<casestmts2>.syn_addr)</casestmts2></casestmts></casestmts>
	free(CASE)
	free( <value>)</value>
	free(COLON)
	free( <statements>)</statements>
	free(BREAK)
	free(SEMICOL)
	free( <casestmts2>)</casestmts2>
130 <casestmts2(1)>→ CASE <value> COLON <statements> BREAK SEMICOL <casestm< td=""><td>nts2(2)&gt;</td></casestm<></statements></value></casestmts2(1)>	nts2(2)>
	<casestmts2(1)>.addr=createNewNode(label:caseValueStmts,<value>.addr,<statements>.syn_addr)</statements></value></casestmts2(1)>
	<casestmts2(1)>.syn_addr=addToStart(<casestmts2(1)>.addr,<casestmts2(2)>.syn_addr)</casestmts2(2)></casestmts2(1)></casestmts2(1)>
	free(CASE)
	free( <value>)</value>
	free(COLON)
	free( <statements>)</statements>
	free(BREAK)
	free(SEMICOL)
	free( <casestmts2(2)>)</casestmts2(2)>
131 <casestmts2>→ ε</casestmts2>	
	<casestmts2>.syn_addr=NULL</casestmts2>
	$free(\epsilon)$
132 <value> → NUM</value>	
	<value>.addr = NUM.addr</value>
133 <value> → TRUE</value>	
	<value>.addr = TRUE.addr</value>
134 <value> → FALSE</value>	
	<value>.addr = FALSE.addr</value>
135 <default_stmt> →DEFAULT COLON <statements> BREAK SEMICOL</statements></default_stmt>	
	<default_stmt>.syn_addr=createNewNode(label:defaultStmt.<statements>.syn_addr)</statements></default_stmt>
	free(DEFAULT)
	free(COLON)
	free( <statements>)</statements>
	free(BREAK)
	free(SEMICOL)
136 <default_stmt> <math>\rightarrow \epsilon</math></default_stmt>	
	<default_stmt>.syn_addr=NULL</default_stmt>
	$free(\epsilon)$
137 <iterativestmt>→ FOR BO ID IN <for_range> BC START <statements> END</statements></for_range></iterativestmt>	
	<iterativestmt>.addr=createNewNode(label:forloop,<for_range>.addr,<statements>.syn_addr)</statements></for_range></iterativestmt>
	free(FOR)
	free(BO)
	free(IN)
	free( <for_range>)</for_range>
	free(BC)
	free(START)
	· · · · · · · · · · · · · · · · · · ·

	free( <statements>)</statements>
	free(END)
138 <iterativestmt>→ WHILE BO <abexpr> BC START <statements> END</statements></abexpr></iterativestmt>	
	<iterativestmt>.addr= createNewNode(label:whileloop,<abexpr>.syn_addr,<statements>.syn_addr)</statements></abexpr></iterativestmt>
	free(WHILE)
	free(BO)
	free( <abexpr>)</abexpr>
	free(BC)
	free(START)
	free( <statements>)</statements>
	free(END)
139 <for_range> → <for_index(1)> RANGEOP <for_index(2)></for_index(2)></for_index(1)></for_range>	
	<for_range>.addr=createNewNode(label:rangeOP,<for_index(1)>.addr,<for_index(2)>.addr)</for_index(2)></for_index(1)></for_range>
	free( <for_index(1)>)</for_index(1)>
	free( <for_index(2)>)</for_index(2)>
	free( <rangeop>)</rangeop>
140 <for_index> → <for_sign> <for_index2></for_index2></for_sign></for_index>	
	<for_index>.addr=createNewNode(label:signedNum,<for_sign>.addr,<for_index2>.addr)</for_index2></for_sign></for_index>
	free( <for_sign>)</for_sign>
	free( <for_index2)< td=""></for_index2)<>
141 <for_index2> → NUM</for_index2>	
	<for_index2>.addr = NUM.addr</for_index2>
142 <for_sign> <math>\rightarrow \epsilon</math></for_sign>	
	<for_sign>.addr=NULL</for_sign>
	$free(\epsilon)$
143 <for_sign> → MINUS</for_sign>	
	<for_sign>.addr=MINUS.addr</for_sign>
144 <for_sign> → PLUS</for_sign>	
	<for_sign>.addr=PLUS.addr</for_sign>