Group No 46		
Name of Team Members: Piklu Paul: 2017A7PS0006P		
Rajababu Saikia: 2017A7PS0007P		
Saurav Virmani: 2017A7PS0090P Siddhant Kharbanda: 2017A7PS0111P		
Sreyas Ravichandran: 2017A7PS01TP Sreyas Ravichandran: 2017A7PS0275P		
Rule	AST Rules	
<pre>section</pre>	program.syn = makenode("PROGRAM",moduleDeclarations.syn, driverModule.syn, otherModules2.syn); otherModules1.inh = moduleDeclaration	
2. <moduledeclarations> "&gt; <moduledeclaration><moduledeclarations> "&gt; <moduledeclarations> "&gt; &lt;</moduledeclarations></moduledeclarations></moduledeclaration></moduledeclarations>	moduleDeclarations2.inh = makenode("MODULE_DECLARATION",moduleDeclarations1.inh, moduleDeclaration.syn); moduleDeclarations1.syn moduleDeclarations.syn = moduleDeclarations.inh	moduleDeclarations2.syn
4. <moduledeclaration> =&gt; DECLARE MODULE ID SEMICOL</moduledeclaration>	moduleDeclaration.syn = makenode("MODULE_DECLARATION", ID.entry)	
5. <othermodules> =&gt; <module><othermodules></othermodules></module></othermodules>	otherModules1.syn = otherModules2.syn; otherModules2.inh = makenode(module.syn, otherModules1.inh); module.inh = otherModules1.inh	
6. <ol> <li><ol> <li>6. <ol> <li>6. <ol></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol></li></ol>	otherModules.syn = otherModules.inh driverDef.syn = makenode("DRIVER_MODULE",moduleDef.syn); moduleDef.inh = driverModule.inh	
8. <module> =&gt; DEF MODULE ID ENDDEF TAKES INPUT SQBO <input_plist> SQBC SEMICOL <ret><moduledef></moduledef></ret></input_plist></module>	module.syn = makenode("MODULE_DEFINITION", ID.entry, input_plist.syn, ret.syn, moduleDef.syn), moduleDef.inh=makenode("MODULE_DEF	,module.inh,input_plist.syn, ret.syn, ID.entry)
9. <rel> ≈&gt; ε 10. <rel> ≈&gt; RETURNS SQBO <output_plist> SQBC SEMICOL</output_plist></rel></rel>	ret.syn = ret.inh ret.syn = output_plist.syn	
11. <input_plist> =&gt; ID COLON <datatype><n1></n1></datatype></input_plist>	input_plist.syn = N1.syn; N1.inh = append(NULL, {ID.entry, dataType.syn}); dataType.inh = input_plist.inh	
12. <n1> =&gt; COMMA ID COLON <datatype> <n1></n1></datatype></n1>	N11.syn = N12.syn; N12.inh = append(N11.inh, {ID.entry, dataType.syn}); dataType.inh = N1.inh	
13. <n1> =&gt; ε  14. <output_plist> =&gt; ID COLON <datatype><n2></n2></datatype></output_plist></n1>	N1.syn = makenode(N1.inh); output_plist.syn = N2.syn; N2.inh = append(NULL, (ID.entry, dataType.syn)); dataType.inh = output_plist.inh	
15. <n2> =&gt; COMMA ID COLON <datatype> <n2></n2></datatype></n2>	N21.syn = N22.syn; N22.inh = append(N21.inh, {ID.entry, dataType.syn}); dataType.inh = N2.inh	
16. <n2> =&gt; €  17. <datatype> =&gt; INTEGER</datatype></n2>	N2.syn = makenode(N2.inh); dataType.syn = makeLeaf("TYPE",INTEGER)	
17. <asia type=""> =&gt; INTEGER  18. <asia type=""> =&gt; REAL</asia></asia>	dataType.syn = makeLeaf("TYPE",REAL)	
19. <datatype> =&gt; BOOLEAN</datatype>	dataType.syn = makeLeaf("TYPE",BOOLEAN)	
20. <datatype> =&gt; ARRAY SQBO <range_arrays> SQBC OF <type> 21. <range_arrays> =&gt; <index> RANGEOP <index></index></index></range_arrays></type></range_arrays></datatype>	dataType.syn = array( range_arrays.syn, type.syn); range_arrays.inh = dataType.inh range_array.syn = makenode("RANGEOP",index1.syn, index2.syn); index1.inh = range_arrays.inh; index2.inh = range_arrays.inh	
22. <type> =&gt; INTEGER</type>	type.syn = makeLeaf("TYPE",INTEGER)	
23. <type> =&gt; REAL 23. <type> =&gt; POOLEAN</type></type>	type.syn = makeLeaf("TYPE",REAL)	
23. <type> =&gt; BOOLEAN 24. <moduledef> =&gt; START <statements> END</statements></moduledef></type>	type.syn = makel.eaf("TYPE",BOOLEAN) moduleDef.syn = statements.syn; statements.inh = moduleDef.inh	
25. <statements> =&gt;<statement> <statements></statements></statement></statements>	statements1.syn = statements2.syn; statements2.inh = makenode("STATEMENTS",statement.syn, statements1.inh); statement.inh = statements	inh
26. <statement> =&gt; ε 27. <statement> =&gt; <iostrnt></iostrnt></statement></statement>	statements.syn = statements.inh statement.syn = ioStmt.syn ; ioStmt.inh = statement.inh	
28. <statement> =&gt; <simplestmt></simplestmt></statement>	statement.syn = simpleStmt.syn; simpleStmt.inh = statement.inh	
29. <statement> =&gt; <declarestmt> 30. <statement> =&gt; <condionalstmt></condionalstmt></statement></declarestmt></statement>	statement.syn = declareStmt.syn; declareStmt.inh = statement.inh	
31. <statement> =&gt; <iterativestmt></iterativestmt></statement>	statement.syn = conditionalStmt.syn; conditionalStmt.inh = statement.inh statement.syn = iterativeStmt.syn; iterativeStmt.inh = statement.inh	
32. <iostmt> =&gt; GET_VALUE BO ID BC SEMICOL</iostmt>	iostmt.syn = makenode( "GET_VALUE", ID.entry)	
33. <lostmt> =&gt; PRINT BO <var> BC SEMICOL 34. <var> =&gt; <pre></pre> <pre>34. <pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></var></var></lostmt>	iostmt.syn = makenode( "PRINT_VALUE", var.syn); var.inh=ioStmt.inh var.syn = boolConstt.syn	
35. <var> =&gt; <var_id_num></var_id_num></var>	var.syn = boulconsat.syn var.syn = var_id_num.syn; var_id_num.inh = var.inh	
36. <boolconstt> =&gt; TRUE 37. <boolconstt> =&gt; FALSE</boolconstt></boolconstt>	boolConstt.syn = makeLeaf("BOOL",TRUE) boolConstt.syn = makeLeaf("BOOL",FALSE)	
38. <var_id_num> =&gt; ID <whichid></whichid></var_id_num>	var_id_num.syn = makeleat("BOUL",FALSE)  var_id_num.syn = makenode("VAR_ID",ID.entry, whichID.syn); whichID.inh = var_id_num.inh	
39. <var_id_num> =&gt; NUM</var_id_num>	var_id_num.syn = makeLeaf("NUM",NUM.val)	
40. <pre></pre> 40. <pre><pre><pre><pre><pre></pre> 41. <pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre>	var_id_num.syn = makeLeaf("RNUM",RNUM.val) whichiD.syn = index.syn; index.inh=whichiD.inh	
42. <whichid> =&gt; s</whichid>	whichID.syn = NULL	
41. <simplestmt> =&gt; <assignmentstmt></assignmentstmt></simplestmt>	simpleStmt.syn = assignmentStmt.syn, assignmentStmt.inh = simpleStmt.inh	
44. <simplestmt> =&gt; <modulereusestmt> 45. <assignmentstmt> =&gt; ID <whichstmt></whichstmt></assignmentstmt></modulereusestmt></simplestmt>	simpleStmt.syn = moduleReuseStmt.syn; moduleReuseStmt.inh = simpleStmt.inh assignmentStmt.syn = whichStmt.syn; whichStmt.inh = ID.entry	
46. <whichstmt> =&gt;<ivalueidstmt></ivalueidstmt></whichstmt>	whichStmt.syn = IvalueIDStmt.syn; IvalueIDstmt.inh = whichStmt.inh	
47. <a equate",lvalueidstmt.inh,="" expression.inh="lvalueIDStmt.inh&lt;/td" expression.syn);="" href="https://www.nching.com/schings-selection-se&lt;/td&gt;&lt;td&gt;whichStmt.syn = lvalueARRStmt.syn; lvalueARRstmt.inh = whichStmt.inh&lt;br&gt;lvalueIDStmt.syn = makenode("><td></td></a>		
49. <\valueARRStmt> => SQBO <\index> SQBC ASSIGNOP <\expression> SEMICOL	IvalueARRStmt.syn = makenode("EQUATE",findArrPos((valueARRStmt.inh, index.syn), expression.syn); index.inh = IvalueARRStmt.inh; expression.syn	on.inh = IvalueARRStmt.inh
50. <index> =&gt; NUM   ID</index>	index.syn = NUM.val   index.syn = ID.entry	on Plant lab
51. <modulereusestmb ==""><optional> USE MODULE ID WITH PARAMETERS <idlist>SEMICOL   52. <optional> =&gt; SQBO <idlist> SQBC ASSIGNOP</idlist></optional></idlist></optional></modulereusestmb>	moduleReuseStmt.syn = makenode("FUNC_CALL",optional.syn, ID.entry, idList.syn); optional.inh = moduleReuseStmt.inh; idList.inh = moduleRe optional.syn = makeNode("RETURN_VALS", idList.syn); idList.inh = optional.inh	iseStmt.inn
54. <optional> ⇒&gt; ε</optional>	optional.syn = NULL	
55. <ldlist> =&gt; ID <n3> 56. <n3> =&gt; COMMA ID <n3></n3></n3></n3></ldlist>	idList.syn = N3.syn; N3.inh = append(NULL, {ID.entry}) N31.syn = N32.syn; N32.inh = append(N31.inh, {ID.entry})	
57. <n3> =&gt; ε</n3>	N3.syn = makenode("ID_LIST", N3.inh);	
58. <expression> =&gt; <arithmeticorbooleanexpr> 58. <expression> =&gt; <u></u></expression></arithmeticorbooleanexpr></expression>	expression.syn = arithmeticOrBooleanExpr.syn; arithmeticOrBooleanExpr.inh = expression.inh	
60. <u>=&gt; <unary op=""> <new nt=""></new></unary></u>	expression.syn = U.syn; U.inh = expression.inh U.syn = makenode("UNARY_EXPR", unary_op.syn, new_NT.syn); new_NT.inh = U.inh	
	new_NT.syn = arithmeticExpr.syn, arithmenticExpr.inh = new_NT.inh	
62. <new_nt> =&gt; <var_id_num></var_id_num></new_nt>	new_NT.syn = var_id_num.syn; var_id_num.inh = new_NT.inh	
62. <new_nt> &gt;&gt; <var.id_num> 63. <unary_op> &gt;&gt; MINUS 64. <unary_op> &gt;&gt; PLUS 64. <unary_op> &gt;&gt; PLUS</unary_op></unary_op></unary_op></var.id_num></new_nt>	new_NT.syn = var_id_num.syn; var_id_num.inh = new_NT.inh unary_op.syn = makehoke("UOp", "-) unary_op.syn = makehoke("UOp", "+)	
62 cnew, NT> => - cnew_d, d_num> 63 cnany_cop => MNUS 64 cnany_cop => PLUS 64 cnany_cop => PLUS 64 cnany_cop => Cnany_cop => - cnay_cop == -	new_NTsyn = var_id_num.pn, var_id_num.inh = new_NT.inh umary_c.psyn = malexholder(Up?, "+) umary_c.psy	rithmeticOrBooleanEupr.linh
62 new /N > > - vera (J, num> 63 v.nery, pp> >> MNUS 64 v.nery, pp> >> PUS 68 v.nermetoC/BooleanEpp> >> -      AnyTerm> - NT> 69 v.nermetoC/BooleanEpp> >> -	new NT-syn = var jd_num_sm; var jd_num.inh = new_NT-inh unary_cp.syn = makeNode(*UOp"; "-) unary_cp.syn = makeNode(*UOp"; "-) arthmetoOlBooleanExpr.syn = NT-syn; NT-inh = makenode("A_OR_BOOL_EXPR": Any Term.syn, arithmetoOlBooleanExpr.inh); Any Term.inh = NT-syn = NT-inh XT-syn = NT-inh XT-syn = NT-inh	rithmeticOrBooleanExpr.inh
62 enew JNT> >> - vent_ d_ num> 63 sunary_0 >> - MNUS 64 vunary_0 >> > PLUS 68 vanary_0 >> - vanyTem> < Nr> 64 vanary_0 >> - vanyTem> < Nr> 64 vanary_0 >> - vanyTem> < Nr> 64 vanary_0 >> - vanyTem> < Nr> 70 vanary - vandquanty - vanary -	new NT-syn = var jd_num.syn var jd_num.syn var jd_num.shn = new NT-inh unary_cp.syn = makeNoder\Upp", 'unary_cp.syn = makeNoder\Upp", 'unary_cp.syn = makeNoder\Upp", 'unary_cp.syn = makeNoder\Upp", 'unary_cp.syn = wtraweloc\Upp", 'unary_cp.syn = wtrawelo	vrithmeticOrBooleanExpr.inh
92 one y.N. ↑ → orar_i.d. num> 3 onan_y.op → MINUS 4 onan_y.op → P.U.S 8 adim/net/ObloanExpr → < - AnyTerm> < - N7> 9 < - N7 → > € 1 onan_y.op → onan_y.op → N7> 1 onan_y.op → onan_y.op → N7> 1 onan_y.op → onan_y.op → onan_y.op → N7> 1 onan_y.op → onan_y.	new NTsyn = var Ld, num.sp., var Ld, num.sp., var Ld, num.sp., var Ld, num.sp., os., pn., meskooder(Up", v) unaty, op. sp. m. makeNoder(Up", v) NT. sp. m. VT. sp., m. vp., m. sp., m. sp.	rithmeticOrBooleanExpr.inh
22 cnew J.M. → svar_i.d., num> 33 sunar_, op → sv MNUS 44 sunar_, op → sv PLUS 86 suntimeted. Debloomerisper → v < ny Term → st7 > 89 st7 - v < 1 - st4 - sunar_, op → sv PLUS 87 st7 - s	new NTsyn = var Lf, num.syn, var Lf, num.syn, var Lf, num.syn, var Lf, num.syn, syn, emeskohed (vp), ".")  unary, cp, syn = malekohed (vp), ".")  unary, cp, syn = malekohed (vp), ".")  unary, cp, syn = malekohed (vp), ".")  nthinselfcollosionalizer syn = NT. syn, NT. inh = malekohed "A, OR_BOOL_EXPR" AnyTerm.syn, arithmeticOrBooleanExpr.inh); AnyTerm.inh = NT. syn = NT. syn, "NT. syn," NT. syn,	rithmeticOrBooleanExpr.inh
62 one W.N. >> - orar_d. d.mm> 3 sunary_op >> DRUS 64 sunary_op >> PLUS 68 suthmeto-OfbooleanExpr >> - < AnyTerm> < NT> 69 < NT> >>   c   c   c   71 < NayTerm> < - < NayTerm> < NT> 71 < NayTerm> >> - c   c   c   72 < NayTerm> < - < c   c   c   73 < NayTerm> < - < c   c   c   74 < NayTerm> < - < c   c   c   75 < NayTerm> < - < c   c   c   76 < NayTerm> < - < c   c   c   77 < NayTerm> < - < c   c   c   78 < NayTerm> < - < c   c   c   79 < NayTerm> < - < c   c   70 < NayTerm> < - < c   c   70 < NayTerm> <	new_NT-syn = var_Lf_num.syn, var_Lf_num.inh = new_NT-inh unary_c pay = makeNobe(PupC, **) ** **NT-syn = NT-sin **NT-syn = NT-syn = NT-sin **Inh	vrithmeticOrBooleanExpr.inh
92. cnew, NT > ≈ vorar_id_num> 3. «nany_op > ≈ PLUS 84. «nany_op > ≈ PLUS 86. «attimate/ObloamExpc × ≈ <any1em> <nt> 89. «NT &gt; ≈ 1  1. «Any1em&gt; ≈ variante/dexpc <nb> 71. «Any1em&gt; ≈ variante/dexpc <nb> 72. «Any1em&gt; ≈ variante/dexpc <nb> 73. «NB » ≈ relationalOp ≈ arithmetic-tpc 74. «NB » ≈ relationalOp ≈ arithmetic-tpc 74. «NB » ≈ relationalOp ≈ settlem × NB&gt; 75. «attimetic-tpc » ≈ stem × NB&gt; 76. «NB » ≈ relationalOp ≈ settlem × NB&gt; 76. «NB » ≈ relationalOp ≈ settlem × NB&gt; 76. «NB » ≈ relationalOp ≈ settlem × NB&gt; 76. «NB » ≈ relationalOp ≈ settlem × NB&gt; 76. «NB » «RB&gt; 76. «NB » «RB&gt; 76. «NB » «RB&gt; 76. «NB» » «RB» » «RB» 76. «NB» »</nb></nb></nb></nt></any1em>	new NTsyn = var Lf, num.syn, var Lf, num.syn, var Lf, num.syn, var Lf, num.syn, syn, emeskohed (vp), ".")  unary, cp, syn = malekohed (vp), ".")  unary, cp, syn = malekohed (vp), ".")  unary, cp, syn = malekohed (vp), ".")  nthinselfcollosionalizer syn = NT. syn, NT. inh = malekohed "A, OR_BOOL_EXPR" AnyTerm.syn, arithmeticOrBooleanExpr.inh); AnyTerm.inh = NT. syn = NT. syn, "NT. syn," NT. syn,	nrithmetic/DisoleanExpr.linh
22. cnew, NT > ≈ vorar_id_num> 35. cnamy_op > ∞ PLUS 84. cnamy_op > ∞ PLUS 86. exitmente/ObloameExpr > ∞ <anyterm <="" nt=""> 89. ≪NT &gt; ∞ i 10. √NT &gt; ∞ califormicExpr &lt; NB&gt; 71. <anyterm> ∞ cariformicExpr &lt; NB&gt; 72. <anyterm> ∞ cariformicExpr &lt; NB&gt; 73. <nb> ∞ relationalOp ≈ cariformicExpr &gt; 1.5 1. <nb> ∞ i 10. √NB &gt; ∞ relationalOp ≈ cariformicExpr &gt; 1.5 1. <nb> ∞ i 10. √NB &gt; ∞ i</nb></nb></nb></anyterm></anyterm></anyterm>	new NTsyn = var Ld, num syn, var Ld, numsuh = new NT-inh unary, op. syn = makeNoder(Up", ") unary, op. syn = NT-ish NT-syn = NT-ish NT-syn = NT-ish NT-syn = NT-syn; NT-izh NT-syn = NT-syn; NT-izh NT-syn = NT-syn; NT-izh AnyTerm syn = NS-syn; NS-inh = makenOder("ARITH_EXPR"-arithmeticExpr.syn, AnyTerm.inh) AnyTerm syn = NS-syn; NS-inh AnyTerm syn = NS-syn; NS-inh = makenOder("ARITH_EXPR"-arithmeticExpr.syn, AnyTerm.inh), arithmeticExpr.inh = AnyTerm.inh NS-syn = arithmeticExpr.syn; arithmeticExpr.inh = makenOder("RELATIONAL_EXPR", relationalOp.syn, NS-inh) NS-syn = NS-syn; NS-inh = makenOder("TERM_EXPR", op 1 syn, Term.syn, NS-inh); term.inh = NS-inh NS-syn = NS-syn; NS-inh = makenOder("TERM_EXPR", op 1 syn, Term.syn, NS-inh); term.inh = NS-inh NS-syn = NS-syn; NS-inh = makenOder("TERM_EXPR", op 1 syn, Term.syn, NS-inh); term.inh = NS-inh	nthmeticOrBooleanExpr.linh
22. cnex // 17 = > cnar i, d // num> 3. cnam / op > > PLUS 44. cnam / op > > PLUS 54. cnam / op > > PLUS 56. cnam / op > > PLUS 56. cnam / op > > < Any Term < \n\frac{1}{17}> 56. cnam / op > > < Any Term < \n\frac{1}{17}> 57. cnam / op > > < Any Term < \n\frac{1}{17}> 57. cnam / op > > < Any Term < \n\frac{1}{17}> 57. cnam / op > \n1	new NT syn + var Lif, num syn, var Liff, num syn,	vrithmeticOrBooleanExpr.inh
22. cnew, NT > ≈ vorar_id_num> 33. cnany_op > ∞ PLUS 84. cnany_op > ∞ PLUS 86. actimated/Collocanicapor × ≈ < AnyTerm > 4NT > 80. 4NT > ≈ 1 7. 4NT > ≈ 1 7. 4NT > ≈ 0 capicalop > 4nyTerm > 4NT > 71. 4AnyTerm > ∞ carithmeticispo > 4NB > 72. 4NPT = ∞ > ∞ bodoconstr 73. 4NB > ∞ creationalop > ∞ carithmeticispo > 1 73. 4NB > ∞ creationalop > ∞ carithmeticispo > 1 74. 4NB > ∞ 1 75. 4nB > ∞ 1 76. 4NB > ∞ 0 77. 4NB > ∞ 1 78. 4NB > ∞ 0 78.	new NTsyn = var Ld, num syn, var Ld, numsuh = new NT-inh unary, op. syn = makeNoder(Up", ") unary, op. syn = NT-ish NT-syn = NT-ish NT-syn = NT-ish NT-syn = NT-syn; NT-izh NT-syn = NT-syn; NT-izh NT-syn = NT-syn; NT-izh AnyTerm syn = NS-syn; NS-inh = makenOder("ARITH_EXPR"-arithmeticExpr.syn, AnyTerm.inh) AnyTerm syn = NS-syn; NS-inh AnyTerm syn = NS-syn; NS-inh = makenOder("ARITH_EXPR"-arithmeticExpr.syn, AnyTerm.inh), arithmeticExpr.inh = AnyTerm.inh NS-syn = arithmeticExpr.syn; arithmeticExpr.inh = makenOder("RELATIONAL_EXPR", relationalOp.syn, NS-inh) NS-syn = NS-syn; NS-inh = makenOder("TERM_EXPR", op 1 syn, Term.syn, NS-inh); term.inh = NS-inh NS-syn = NS-syn; NS-inh = makenOder("TERM_EXPR", op 1 syn, Term.syn, NS-inh); term.inh = NS-inh NS-syn = NS-syn; NS-inh = makenOder("TERM_EXPR", op 1 syn, Term.syn, NS-inh); term.inh = NS-inh	rithmetic/OBooleanExpr.inh
22. dney JN = ∞ var_i Jd_num> 33. sunan_ op ∞ w MNUS 44. sunan_ op ∞ w PLUS 84. sunan_ op ∞ w PLUS 85. suntimeted Delosenistique ∞ < Any Terms < NT> 85. staffwreted Delosenistique ∞ < Any Terms < NT> 85. staffwreted Delosenistique ∞ < Any Terms < NT> 71. sk. y Flag = NT	new NTsyn = var Lf, num syn, var Lf, numsinh = new NT-inh unary, op. syn = makeNoder(UpC), "-1) unary, op. syn = wid widen (upC), "-1) unary, op. syn = vXT-syn, "NT-inh", "nyn NT-inh", "nyn NT-inh", "nyn "NT-inh",	rithmeticOrBooleanExpr.inh
92 one y.M. > > orar_i.d. num> 3 sunar_j.op> > P.LUS 64 sunar_j.op> > P.LUS 66 sunar_j.op> > > P.LUS 68 sunar_j.op> > > > > > P.LUS 68 sunar_j.op> > > > > > > > > > > > > > > > > > >	new NT syn = var_Is_num.syn, var_Is_num.syn, var_Is_num.inh = new_NT.inh unary_c pp, on = makeNobe(Pup*, "+") inh =	vrithmeticOrBooleanExpr.inh
92 one y.N. ↑ = > - vari_d.mm> 3 vani_y.go > > P.U.S 64 vani_y.go > > P.U.S 68 vani_y.go > > P.U.S 68 vani_y.go > > - vany1em> - vi7> 68 vi7. > > 1 7 vi7. > > 1 7 vi7. > > 1 7 vi7. > - vi7 vi7. > -	new NT syn = var_Id_num.syn, var_Id_num.syn, var_Id_num.syn = new_NT.inb unary_c.psyn = makeNobe(PUp)* "> unary_c.psyn = nt/Inb NT.syn = NT.inb NT.syn = NT.inb NT.syn = NT.syn NT.inb = makeNobe(PUDICAL_EXPR*.logicalCp.syn, NT.inb, AnyTerm.syn). Anyterm.inh = NT.inb AnyTerm.syn = NS.syn, NB.inb = makeNobe(PITELTH_EXPR*_arithmeticExpr.syn, AnyTerm.inb); arithmeticExpr.inh = AnyTerm.inh AnyTerm.syn = boolconsist.syn: NB.syn = arithmeticExpr.syn, arithmeticExpr.inh = makeNobe(PITELATIONAL_EXPR*_relationaDp.syn, NB.inb) NB.syn = nB.sinh simmeticExpr.syn = NB.syn, NB.inb = makeNobe(PITELATIONAL_EXPR*_relationaDp.syn, NB.inb) NB.syn = NB.sinh simmeticExpr.syn = NB.syn, NB.inb = makeNobe(PITENLEXPR*_cop1.syn, term.syn, arithmeticExpr.inb) NB.syn = NB.sinh simmeticExpr.syn = NB.syn, NB.sinh = makeNobe(PITENLEXPR*_cop1.syn, term.syn, NB.sinh); learn.inb = NB.sinh simmeticExpr.syn = NB.syn, NB.sinh = makeNobe(PITENLEXPR*_cop1.syn, term.syn, NB.sinh); learn.inb = NB.sinh simmeticExpr.syn = NB.sinh simmeticExpr.syn = NB.sinh simmeticExpr.syn = NB.syn, NB.sinh = makeNobe(PITENLEXPR*_cop1.syn, term.syn, NB.sinh); learn.inb = NB.sinh simmeticExpr.syn	rithmetic/DiscleanExpr.inh
22. cnew, NT > ≈ vorar_id_num> 33. chany_op > ∞ PLUS 84. chany_op > ∞ PLUS 86. ***chimetelColores of per volume ***chiyTem* · «NT > 89. ***NT > ∞ t 89. **NT > ∞ t 99. **NT > ∞ t 10. ***NT > ∞ choolores of per volume ***NT > 11. ***AnyTem* > ∞ carithmeticExpr · «NB> 12. **AnyTem* > ∞ carithmeticExpr · «NB> 13. **NB > ∞ creationalOp ~ carithmeticExpr > 13. **NB > ∞ creationalOp ~ carithmeticExpr > 15. ***carithmeticExpr > ∞ stem* · «NB> 17. ***ANB > ∞ t 18. ***ANB > ∞ t 18. ***ANB > ∞ carithmeticExpr > ∞ stem* · «NB> 19. **NB > ∞ carithmeticExpr > ∞ stem* · «NB> 19. **NB > ∞ carithmeticExpr > ∞ stem* · «NB> 19. **NB > ∞ carithmeticExpr > ∞ stem* · «NB> 19. ***ANB > ∞ carithmeticExpr > ∞ stem* · «NB> 19. ***ANB > ∞ carithmeticExpr > ∞ stem* · «NB> 19. **Stem* > ∞ catchr · «NB>	new NTsyn = var Ld, num.syn, var Ld, num.syn, var Ld, num.syn, var Ld, num.syn, os. yn emsekhoder (Upo; "-") unary, op. syn = makehoder (Upo; "-") unary, op. syn = war. syn N.	rithmeticOrBooleanEspr.inh
22. cnew JNT → so var Jd Jnum> 33. cnamy_op > PLUS 44. cnamy_op > PLUS 54. cnamy_op > PLUS 56. cnamy_op > PLUS 57. cnamy_op >	new NT syn = var_Lif_num.syn, var_Lif_num.syn var_Lif_num.syn = new_NT.inh unary_c pay = makeNobe(PUpC) *** ** ** ** ** ** ** ** ** ** ** ** **	vrithmeticOrBooleanExpr.inh
92 one y.N. ↑ = > - orar_i.d. num> 3 orar_j.op > > P.U.S 64 orar_j.op > > P.U.S 68 orar_j.op > > P.U.S 68 orar_j.op > > P.U.S 68 orar_j.op > > P.U.S 69 orar_j.op > > - orar_j.op > > Orar_j.op > Orar	new NT syn = var Lif, num syn, var Lift syn, var L	inthmeticOfBooleanExpr.inh
92. cnex JM. Tr ⇒ vora Jd. num> 3. surany_op > PLUS 44. surany_op > PLUS 54. surany_op > PLUS 56. surany_op > PLU	new NT syn = var_Lif_num.syn, var_Lif_num.syn var_Lif_num.syn = new_NT.inh unary_c pay = makeNobe(PUpC) *** ** ** ** ** ** ** ** ** ** ** ** **	rithmeticOrBooleanEspr.inh
22. cnex J.M. = 9 cnex J.d. jumn> 33. cnany_op = 9 PLUS 44. custony_op = 9 PLUS 54. custony_op = 9 Cu	new NT syn = var_Lit_num.syn, var_Lit_num.syn, var_Lit_num.syn = new_NT.inh unary, op. pn = makeNobe(PupC) **' in = new nobe(PupC)	vrithmeticOrBooleanExpr.inh
92. cnew, NT > ≈ votar_id_num> 3. cnamp_op > PLUS 84. cnamp_op > PLUS 84. cnamp_op > PLUS 86. vdtm/emc0foolomefape > v < AnyTerm < NT > 86. vdt. > v < Colorate	new NT syn = var_ld_num.syn, var_ld_num.syn var_ld_num.syn = new_NT.inb uraty_c.pp.syn = makeNobe(*UpC)* "> uraty_c.pp.syn = NT.inb NT syn = NT.inb AnyTerm.syn = NB syn; NB.inb = makeNode(*UpC)* "A arithmeticExpr.syn, AnyTerm.inb); arithmeticExpr.inb = AnyTerm.syn = Nobicontait.syn; NB.syn = arithmeticExpr.syn; arithmeticExpr.inb = makeNode(*TELATIONAL_EXPR**, relationation_psyn, NB.inb) NB.syn = NB.inb arithmeticExpr.syn = NB.syn; NB.inb = makeNode(*TERM_EXPR**, opt.syn, term.syn, arithmeticExpr.inb) NB.syn = NB.inb At syn = NB.inb arithmeticExpr.syn; arithmeticExpr.inb NB.syn = NB.inb NB.syn = NB.inb inb inb inb inb inb inb inb inb inb	inthmeticOrBooleanExpr.inh
22. cnew, NT = ∞ vari_d nume 3. clann_gop ≈ PLUS 64. clann_gop ≈ PLUS 66. vdnran_gop ≈ PLUS 66. vdnran_gop ≈ PLUS 66. vdnran_gop ≈ vdnran_cdra	Internal Name var Lif, num. sp., var Liften St., va	
22. cnew, NT = ∞ vari_ M_num> 33. cnam_qop > ∞ PLUS 44. cnam_qop > ∞ PLUS 54. cnam_qop > ∞ PLUS 56. cnam_pop > ∞ PLUS 56. vari_ va	Inter. NT syn = var_Lif_num.syn, var_Lif_num.syn, var_Lif_num.syn = new_NT-linb unary_c pp, or makeNobe(PUp; "") unary_c pp, or NT_cinb NT-syn = NT_cinb NT_syn = NT_cinb NT_cinb NT_syn = NT_cinb N	
22 cnew J.M. → so vari_J.d., num> 33 sunar_yop → S. PLUS 84 sunar_yop → S. PLUS 85 sunar_yop → S. PLUS 86 sunar_yop → S. PLUS 87 sunar_yop → S. PLUS 88 sunar_yop → S. PLUS 89	Internal Name var Lif, num. sp.,	
22. cnex JNT = 9 cnex Jd Jnum> 33. cnamy_op > 9 FLUS 44. cnamy_op > 9 FLUS 54. cnamy_op > 9 FLUS 56. cnamy_op > 9 FLUS 57. cnamy_op	Inex. NT syn = var_Is_num.syn, var_Is_num.syn var_Is_num.inh = new_NT.inh unary_c pay = makeNobe(PUpC) ***  unary_c pay = NT.inh NT.syn = NT.syn NT.inh = makeNobe(PUpC) ***  unary_c pay = NT.inh NT.syn = NT.syn NT.inh = makeNobe(PUpC) ***  unary_c pay = NT.inh NT.syn = NT.syn NT.inh = makeNobe(PUpC) ***  unary_c pay = NT.inh NT.syn = NT.syn NT.inh = makeNobe(PUpC) ***  unary_c pay = NT.inh NT.syn = NT.syn NT.inh = makeNobe(PUpC) ***  unary_c pay = nT.inh ***	
22. cnew, NT = ∞ vari_d num> 3. cnam_gop ≈ PLUS 44. cnam_gop ≈ PLUS 56. cntfmetoColosentExpr ≈ v < AnyTerm < NT > 56. cntfmetoColosentExpr ≈ v < AnyTerm < NT > 56. cntfmetoColosentExpr ≈ v < AnyTerm < NT > 57. cnt	new_NTsys = var_is_nums_yn, var_is_nums_nv var_is_numinh = new_NT.inh unary_c psys = makeNobe(PupCy, **) unary_c psys = new_NT.inh NT_sys = NT_sinh NT_sinh NT_sys = NT_sinh	
22. cnear, VPT ⇒ vorar, d., numo 32. cnamy, op ⇒ VPLUS 43. cnamy, op ⇒ VPLUS 44. cnamy, op ⇒ VPLUS 54. cnamy, op ⇒ VPLUS 55. cnamy, op ⇒ VPLUS 65. cnamy	Inex. NT syn = var_Is_num.syn, var_Is_num.syn var_Is_num.inh = new_NT.inh unary_c pay = makeNobe(PUpC) ***  unary_c pay = NT.inh NT.syn = NT.syn NT.inh = makeNobe(PUpC) ***  unary_c pay = NT.inh NT.syn = NT.syn NT.inh = makeNobe(PUpC) ***  unary_c pay = NT.inh NT.syn = NT.syn NT.inh = makeNobe(PUpC) ***  unary_c pay = NT.inh NT.syn = NT.syn NT.inh = makeNobe(PUpC) ***  unary_c pay = NT.inh NT.syn = NT.syn NT.inh = makeNobe(PUpC) ***  unary_c pay = nT.inh ***	
62. cmany. yp >> PLUS 63. cmany. yp >> PLUS 64. cmany. yp >> PLUS 64. cmany. yp >> PLUS 65. cmany. yp >> PLUS 66. cmany. yp >> PLUS 66. cmany. yp >> PLUS 67. cmany. yp >> PLUS 68. cmany. yp >> PLUS 69. cmany. yp >> PLUS	Inex. NT sys = var_Ls_num.syn, var_Ls_num.syn, var_Ls_num.sh = new_NT.inb unary_c pays = makeNobe(PupCy **) unary_c pays = var_unary_c pays = var_unary	
61. classy, NT >=> 8D - Cartifronte(EEXpp- BC	Inew NT sys = var_ls_num.syn, var_ls_num.syn, var_ls_num.sh = new_NT.inb unary_c.ps_pri = makeNobe(PUp; **) unary_c.ps_pri = not syn_nt_num.sh = n	
62. cmany. yp >> PLUS 63. cmany. yp >> PLUS 64. cmany. yp >> PLUS 64. cmany. yp >> PLUS 65. cmany. yp >> PLUS 66. cmany. yp >> PLUS 66. cmany. yp >> PLUS 67. cmany. yp >> PLUS 68. cmany. yp >> PLUS 69. cmany. yp >> PLUS	Inex. NT sys = var_Ls_num.syn, var_Ls_num.syn, var_Ls_num.sh = new_NT.inb unary_c pays = makeNobe(PupCy **) unary_c pays = var_unary_c pays = var_unary	ich