# INFO-F-403: Language theory and compiling Rapport projet partie 2 - Grammaire

Simon Picard Arnaud Rosette

18 décembre 2014

# 1 Transformation de la grammaire donnée

### 1.1 Opérateurs binaires et unaires

La première étape pour rendre la grammaire LL(1) fut la distinction des deux types d'opérateurs : les opérateurs binaires qui agissent sur deux expressions l'une située à gauche et l'autre à droite de ces opérateurs et les opérateurs unaires qui agissent sur une seule expression située à droite de ces opérateurs. Il faut différencier les deux types d'opérateurs afin de ne pas avoir des expressions du style : \*4, \*2, 6|,... Les opérateurs unaires sont au nombre de quatre :! (negation),  $\sim$  (not bit à bit), + et -. Les opérateurs + et - sont également des opérateurs binaires.

## 1.2 Priorité et associativité des opérateurs

Lors de cette étape, nous avons modifié la grammaire afin de fixer la priorité et l'associativité des différents opérateurs, ceci afin de rendre la grammaire moins ambigue. Nous avons remarqué que les opérateurs unaires étaient plus prioritaires que les opérateurs binaires et que les opérateurs unaires étaient tous associatifs à droite, tandis que les opérateurs binaires l'étaient tous à gauche. Pour fixer la priorité, nous avons mis les opérateurs les moins prioritaires le plus "haut" possible dans la grammaire et les plus prioritaires le plus "bas" possible.

## 1.3 Suppression des récursion à gauche

L'étape suivante fut l'étape de suppression des récursions à gauche qui sont incompatibles avec les "top-down parser" car cela fait entrer ce genre de parser dans une boucle ou récursion infinie.

#### 1.4 Left factoring

Cette étape rassemble les règles de production d'une variable qui ont un préfixe commun en une seule règle de production qui contient ce préfixe commun et une nouvelle variable. Cette nouvelle variable possède des règles de production vers les différents suffixes qui étaient présents à l'origine dans les différentes règles de production qui ont été mises en commun. Ceci est nécessaire car le parser qu'on va créer est LL(1), il faut donc qu'il puisse choisir la bonne règle de production en regardant seulement le prochain token qui est en entrée.

# 1.5 Suppression de la variable <Instruction>

Nous avons décidé de supprimer la variable <Instruction> et de ne laisser que <InstructionList>. <InstructionList> pouvant être une liste d'instruction, une instruction vide ou rien (epsilon), ceci afin de rendre la grammaire moins ambigue, plus allégée et sans grande perte de fonctionnalité du langage.

### 2 Grammaire

- $\begin{array}{ccc} [1] & < \operatorname{Program} > & \rightarrow & < \operatorname{InstructionList} > \end{array}$
- [2] <Instruction>  $\rightarrow$  <IdentifierInstruction>

[3]		$\rightarrow$	<constdefinition></constdefinition>
[4]		$\stackrel{'}{ ightarrow}$	<block></block>
[5]		$\rightarrow$	<loop></loop>
[6]		$\rightarrow$	<builtinfunctioncall></builtinfunctioncall>
[7]		$\rightarrow$	<functiondefinition></functiondefinition>
[8]	<InstructionList $>$	$\stackrel{'}{ ightarrow}$	<instruction> <instructionlisttail></instructionlisttail></instruction>
[9]	(IIIS) ucololisis	$\stackrel{'}{ ightarrow}$	<instructionlisttail></instructionlisttail>
[10]	<instructionlisttail></instructionlisttail>	$\stackrel{'}{ ightarrow}$	END OF INSTRUCTION <instructionlist></instructionlist>
[11]		$\stackrel{'}{ ightarrow}$	$\epsilon$
[12]	<IdentifierInstruction $>$	$\stackrel{'}{\rightarrow}$	IDENTIFIER <identifierinstructiontail></identifierinstructiontail>
[13]	<identifierinstructiontail></identifierinstructiontail>	$\stackrel{'}{ ightarrow}$	<assignationtail></assignationtail>
[14]	\identification ran	$\stackrel{}{\rightarrow}$	TYPE_DEFINITION <type></type>
[15]		$\stackrel{'}{ ightarrow}$	<functioncalltail></functioncalltail>
[16]	<assignationtail></assignationtail>	$\stackrel{'}{ ightarrow}$	ASSIGNATION < Expression >
[17]	(Tissighation Tail)	$\stackrel{'}{\rightarrow}$	COMMA IDENTIFIER <assignationtail> COMMA</assignationtail>
[11]		,	<expression></expression>
[18]	<constdefinition></constdefinition>	$\rightarrow$	CONST IDENTIFIER <assignationtail></assignationtail>
[19]	<block></block>	$\rightarrow$	LET IDENTIFIER <assignationtail></assignationtail>
[19]	\Diock >	$\rightarrow$	END OF INSTRUCTION <instructionlist> END</instructionlist>
[20]	<loop></loop>	$\rightarrow$	<if></if>
[20]	\L00p>	$\rightarrow$	WHILE <expression> END OF INSTRUCTION</expression>
[21]		$\rightarrow$	<pre><instructionlist> END</instructionlist></pre>
[22]		$\rightarrow$	FOR IDENTIFIER ASSIGNATION < Expression>
		$\rightarrow$	TERNARY ELSE <expression> <fortail></fortail></expression>
[23]	<fortail></fortail>	,	END_OF_INSTRUCTION <instructionlist> END</instructionlist>
	< FOI Tall>	$\rightarrow$	TERNARY ELSE <expression></expression>
[24]		$\rightarrow$	
[05]	<true o=""></true>	,	END_OF_INSTRUCTION <instructionlist> END</instructionlist>
[25]	<Type $>$	$\rightarrow$	BOOLEAN_TYPE
[26]		$\rightarrow$	REAL_TYPE
[27]	<	$\rightarrow$	INTEGER_TYPE
[28]	<expression></expression>	$\rightarrow$	<pre><binaryexpression> <ternaryifexpression></ternaryifexpression></binaryexpression></pre>
[29]	<TernaryIfExpression $>$	$\rightarrow$	TERNARY_IF <expression></expression>
[00]		,	<ternaryelseexpression></ternaryelseexpression>
[30]		$\rightarrow$	E CERTAIN FILES OF A STATE OF A S
[31]	<ternaryelseexpression></ternaryelseexpression>	$\rightarrow$	TERNARY_ELSE < Expression >
[32]	<AtomicExpression $>$	$\rightarrow$	<atomicidentifierexpression></atomicidentifierexpression>
[33]		$\rightarrow$	INTEGER
[34]		$\rightarrow$	REAL
[35]		$\rightarrow$	BOOLEAN
[36]	A. TI C. T	$\rightarrow$	<builtinfunctioncall></builtinfunctioncall>
[37]	$<\!$	$\rightarrow$	IDENTIFIER
[0.0]	A IIIC D		<atomicidentifierexpressiontail></atomicidentifierexpressiontail>
[38]	$<\!$	$\rightarrow$	<functioncalltail></functioncalltail>
[39]		$\rightarrow$	€ NEGATION II E
[40]	<UnaryExpression $>$	$\rightarrow$	NEGATION <unaryexpression></unaryexpression>
[41]	II. Du I N I	$\rightarrow$	<unarybitwisenotexpression></unarybitwisenotexpression>
[42]	<UnaryBitwiseNotExpression $>$	$\rightarrow$	BITWISE_NOT <unarybitwisenotexpression></unarybitwisenotexpression>
[43]		$\rightarrow$	<unaryminusplusexpression></unaryminusplusexpression>
[44]	<UnaryMinusPlusExpression $>$	$\rightarrow$	MINUS <unaryminusplusexpression></unaryminusplusexpression>
[45]		$\rightarrow$	PLUS <unaryminusplusexpression></unaryminusplusexpression>
[46]		$\rightarrow$	<UnaryAtomicExpression $>$
[47]	<UnaryAtomicExpression $>$	$\rightarrow$	<atomicexpression></atomicexpression>
[48]		$\rightarrow$	${\tt LEFT\_PARENTHESIS} < \!$
			RIGHT_PARENTHESIS
[49]	<binaryexpression></binaryexpression>	$\rightarrow$	<binarylazyorexpression></binarylazyorexpression>
			<binaryexpression'></binaryexpression'>
[50]	<BinaryExpression' $>$	$\rightarrow$	$LAZY\_OR < BinaryLazyOrExpression >$
			<binaryexpression'></binaryexpression'>

[ <del>-</del> 1			
[51]		$\rightarrow$	$\epsilon$
[52]	$<\!$	$\rightarrow$	<BinaryLazyAndExpression $>$
			<BinaryLazyOrExpression' $>$
[53]	<BinaryLazyOrExpression' $>$	$\rightarrow$	$LAZY\_AND < BinaryLazyAndExpression >$
			<binarylazyorexpression'></binarylazyorexpression'>
[54]		$\rightarrow$	$\epsilon$
[55]	<binarylazyandexpression></binarylazyandexpression>	$\rightarrow$	<binarynumericexpression></binarynumericexpression>
[00]	(Billary Baby Fillabilip Fossion)	,	<binarylazyandexpression'></binarylazyandexpression'>
[56]	<binarylazyandexpression'></binarylazyandexpression'>	$\rightarrow$	GREATER THAN <binarynumericexpression></binarynumericexpression>
[90]	\Dinary Lazy MidExpression >	/	<pre><binarylazyandexpression'></binarylazyandexpression'></pre>
[57]		,	LESS THAN <binarynumericexpression></binarynumericexpression>
[57]		$\rightarrow$	_
[=0]			<binarylazyandexpression'> CREATER OR FOLIAL CONTRACT</binarylazyandexpression'>
[58]		$\rightarrow$	GREATER_OR_EQUALS_THAN
			<binarynumericexpression></binarynumericexpression>
			<binarylazyandexpression'></binarylazyandexpression'>
[59]		$\rightarrow$	LESS_OR_EQUALS_THAN
			<BinaryNumericExpression $>$
			<binarylazyandexpression'></binarylazyandexpression'>
[60]		$\rightarrow$	EQUALITY <binarynumericexpression></binarynumericexpression>
			<binarylazyandexpression'></binarylazyandexpression'>
[61]		$\rightarrow$	INEQUALITY <binarynumericexpression></binarynumericexpression>
. ,			<binarylazyandexpression'></binarylazyandexpression'>
[62]		$\rightarrow$	$\epsilon$
[63]	<binarynumericexpression></binarynumericexpression>	$\rightarrow$	<binarytermexpression></binarytermexpression>
[oo]	(Binary) (americExpression)	,	<binarynumericexpression'></binarynumericexpression'>
[64]	<binarynumericexpression'></binarynumericexpression'>	$\rightarrow$	PLUS <binarytermexpression></binarytermexpression>
[04]	\DinaryivumericExpression >	7	<pre><binarynumericexpression'></binarynumericexpression'></pre>
[6]		,	MINUS <binarytermexpression></binarytermexpression>
[65]		$\rightarrow$	· -
[00]			<binarynumericexpression'></binarynumericexpression'>
[66]		$\rightarrow$	BITWISE_OR <binarytermexpression></binarytermexpression>
[ o = ]			<binarynumericexpression'></binarynumericexpression'>
[67]		$\rightarrow$	BITWISE_XOR <binarytermexpression></binarytermexpression>
			<binarynumericexpression'></binarynumericexpression'>
[68]		$\rightarrow$	$\epsilon$
[69]	$<\!$	$\rightarrow$	<BinaryShiftedExpression $>$
			<BinaryTermExpression' $>$
[70]	<BinaryTermExpression' $>$	$\rightarrow$	ARITHMETIC_SHIFT_LEFT
			<binaryshiftedexpression></binaryshiftedexpression>
			<binarytermexpression'></binarytermexpression'>
[71]		$\rightarrow$	ARITHMETIC SHIFT RIGHT
. ,			<binaryshiftedexpression></binaryshiftedexpression>
			<binarytermexpression'></binarytermexpression'>
[72]		$\rightarrow$	$\epsilon$
[73]	<BinaryShiftedExpression $>$	$\overset{'}{ ightarrow}$	<binaryfactorexpression></binaryfactorexpression>
[10]	(Binary Simrocatinprossion)	,	<binaryshiftedexpression'></binaryshiftedexpression'>
[74]	<binaryshiftedexpression'></binaryshiftedexpression'>	$\rightarrow$	TIMES <binaryfactorexpression></binaryfactorexpression>
[14]	\DinarySimtedExpression >	7	<pre><binaryshiftedexpression'></binaryshiftedexpression'></pre>
[75]		,	DIVIDE <binaryfactorexpression></binaryfactorexpression>
[75]		$\rightarrow$	
[=0]			<pre><binaryshiftedexpression'></binaryshiftedexpression'></pre>
[76]		$\rightarrow$	REMAINDER <binaryfactorexpression></binaryfactorexpression>
F1			<binaryshiftedexpression'></binaryshiftedexpression'>
[77]		$\rightarrow$	BITWISE_AND <binaryfactorexpression></binaryfactorexpression>
			<BinaryShiftedExpression' $>$
[78]		$\rightarrow$	$INVERSE\_DIVIDE < BinaryFactorExpression>$
			<BinaryShiftedExpression' $>$
[79]		$\rightarrow$	$\epsilon$
[80]	$<\!$	$\rightarrow$	<UnaryExpression $>$
- •	_		<binaryfactorexpression'></binaryfactorexpression'>

[04]			DOMED II I
[81]	<BinaryFactorExpression' $>$	$\rightarrow$	POWER < Unary Expression >
[00]			<BinaryFactorExpression' $>$
[82]	ATC:	$\rightarrow$	E END OF INCEDITORION
[83]	<if></if>	$\rightarrow$	IF <expression> END_OF_INSTRUCTION</expression>
[0.4]	.ICD 1.		<instructionlist> <ifend></ifend></instructionlist>
[84]	<ifend></ifend>	$\rightarrow$	ELSE_IF <expression> END_OF_INSTRUCTION</expression>
[0 <b>=</b> ]			<instructionlist> <ifend></ifend></instructionlist>
[85]		$\rightarrow$	ELSE <instructionlist> END</instructionlist>
[86]	5 6	$\rightarrow$	END
[87]	<BuiltInFunctionCall $>$	$\rightarrow$	READ_REAL LEFT_PARENTHESIS
r 1			RIGHT_PARENTHESIS
[88]		$\rightarrow$	READ_INTEGER LEFT_PARENTHESIS
r1			RIGHT_PARENTHESIS
[89]		$\rightarrow$	INTEGER_CAST LEFT_PARENTHESIS <expression></expression>
			RIGHT_PARENTHESIS
[90]		$\rightarrow$	REAL_CAST LEFT_PARENTHESIS < Expression >
			RIGHT_PARENTHESIS
[91]		$\rightarrow$	BOOLEAN_CAST LEFT_PARENTHESIS < Expression >
			RIGHT_PARENTHESIS
[92]		$\rightarrow$	PRINTLN LEFT_PARENTHESIS < Expression>
			RIGHT_PARENTHESIS
[93]	<FunctionCallTail $>$	$\rightarrow$	LEFT_PARENTHESIS < Parameter>
			RIGHT_PARENTHESIS
[94]	<parameter></parameter>	$\rightarrow$	<Expression $>$ $<$ ParameterTail $>$
[95]		$\rightarrow$	$\epsilon$
[96]	<ParameterTail $>$	$\rightarrow$	${\rm COMMA} < \!$
[97]		$\rightarrow$	$\epsilon$
[98]	<FunctionDefinition $>$	$\rightarrow$	FUNCTION IDENTIFIER LEFT_PARENTHESIS
			$<$ Argument $>$ RIGHT_PARENTHESIS
			$<\!\!\text{InstructionList}\!\!><\!\!\text{FunctionDefinitionEnd}\!\!>$
[99]	<FunctionDefinitionEnd $>$	$\rightarrow$	RETURN < Expression > END
[100]		$\rightarrow$	END
[101]	<Argument $>$	$\rightarrow$	IDENTIFIER TYPE_DEFINITION < Type>
			<argumenttail></argumenttail>
[102]		$\rightarrow$	$\epsilon$
[103]	<ArgumentTail $>$	$\rightarrow$	COMMA IDENTIFIER TYPE_DEFINITION < Type>
- •			<argumenttail></argumenttail>
[104]		$\rightarrow$	$\epsilon$

# 3 Ensembles First et Follow

Variable	First	Follow
	FUNCTION, WHILE, READ_REAL	
	EPSILON_VALUE	
	IDENTIFIER, CONST	
<program></program>	BOOLEAN_CAST, PRINTLN	
< r rogram>	END_OF_INSTRUCTION	
	READ_INTEGER, FOR	
	INTEGER_CAST, LET, IF	
	$\overline{ ext{REAL}}$ _CAST	
	FUNCTION, READ_INTEGER	
	FOR, WHILE, READ_REAL	
<instruction></instruction>	INTEGER_CAST	END, END_OF_INSTRUCTION
< mstruction>	BOOLEAN_CAST, CONST	ELSE_IF, ELSE, RETURN
	IDENTIFIER, PRINTLN, LET, IF	
	REAL_CAST	

<InstructionList $>$	FUNCTION, WHILE, READ_REAL EPSILON_VALUE BOOLEAN_CAST, IDENTIFIER CONST, PRINTLN END_OF_INSTRUCTION READ_INTEGER, FOR INTEGER_CAST, LET, IF REAL_CAST	END, ELSE_IF, ELSE, RETURN
<instructionlisttail></instructionlisttail>	EPSILON_VALUE END_OF_INSTRUCTION	${\tt END,ELSE\_IF,ELSE,RETURN}$
<identifierinstruction></identifierinstruction>	IDENTIFIER	END, END_OF_INSTRUCTION ELSE_IF, ELSE, RETURN
<identifierinstructiontail></identifierinstructiontail>	ASSIGNATION TYPE_DEFINITION, COMMA LEFT_PARENTHESIS	END, END_OF_INSTRUCTION ELSE_IF, ELSE, RETURN
<assignationtail></assignationtail>	ASSIGNATION, COMMA	END, COMMA END_OF_INSTRUCTION ELSE_IF, ELSE, RETURN
<constdefinition></constdefinition>	CONST	END, END_OF_INSTRUCTION ELSE_IF, ELSE, RETURN
<block></block>	LET	END, END_OF_INSTRUCTION ELSE_IF, ELSE, RETURN
<loop></loop>	FOR, WHILE, IF	END, END_OF_INSTRUCTION ELSE_IF, ELSE, RETURN
<fortail></fortail>	END_OF_INSTRUCTION TERNARY_ELSE	END, END_OF_INSTRUCTION ELSE_IF, ELSE, RETURN
<type></type>	REAL_TYPE, BOOLEAN_TYPE INTEGER_TYPE	RIGHT_PARENTHESIS, END COMMA END_OF_INSTRUCTION ELSE_IF, ELSE, RETURN
<expression></expression>	READ_INTEGER, INTEGER REAL, PLUS, BOOLEAN NEGATION, MINUS BITWISE_NOT, READ_REAL INTEGER_CAST LEFT_PARENTHESIS BOOLEAN_CAST, IDENTIFIER PRINTLN, REAL_CAST	RIGHT_PARENTHESIS, END COMMA END_OF_INSTRUCTION ELSE_IF, ELSE TERNARY_ELSE, RETURN
<ternaryifexpression></ternaryifexpression>	TERNARY_IF EPSILON_VALUE	RIGHT_PARENTHESIS, END COMMA END_OF_INSTRUCTION ELSE_IF, ELSE TERNARY_ELSE, RETURN
<ternaryelseexpression></ternaryelseexpression>	TERNARY_ELSE	RIGHT_PARENTHESIS, END COMMA END_OF_INSTRUCTION ELSE_IF, ELSE TERNARY_ELSE, RETURN

<AtomicExpression $>$	READ_INTEGER, INTEGER REAL, BOOLEAN, READ_REAL INTEGER_CAST BOOLEAN_CAST, IDENTIFIER PRINTLN, REAL_CAST	ARITHMETIC_SHIFT_RIGHT EQUALITY, BITWISE_AND ARITHMETIC_SHIFT_LEFT TERNARY_IF, GREATER_THAN LESS_OR_EQUALS_THAN ELSE, TERNARY_ELSE, POWER INEQUALITY, BITWISE_OR MINUS, RIGHT_PARENTHESIS BITWISE_XOR, REMAINDER COMMA END_OF_INSTRUCTION RETURN, LESS_THAN LAZY_AND, PLUS, LAZY_OR INVERSE_DIVIDE, END, TIMES ELSE_IF, DIVIDE GREATER_OR_EQUALS_THAN
<AtomicIdentifierExpression $>$	IDENTIFIER	ARITHMETIC_SHIFT_RIGHT EQUALITY, BITWISE_AND ARITHMETIC_SHIFT_LEFT TERNARY_IF, GREATER_THAN LESS_OR_EQUALS_THAN ELSE, TERNARY_ELSE, POWER INEQUALITY, BITWISE_OR MINUS, RIGHT_PARENTHESIS BITWISE_XOR, REMAINDER COMMA END_OF_INSTRUCTION RETURN, LESS_THAN LAZY_AND, PLUS, LAZY_OR INVERSE_DIVIDE, END, TIMES ELSE_IF, DIVIDE GREATER OR EQUALS THAN
< Atomic Identifier Expression Tail>	EPSILON_VALUE LEFT_PARENTHESIS	ARITHMETIC_SHIFT_RIGHT EQUALITY, BITWISE_AND ARITHMETIC_SHIFT_LEFT TERNARY_IF, GREATER_THAN LESS_OR_EQUALS_THAN ELSE, TERNARY_ELSE, POWER INEQUALITY, BITWISE_OR MINUS, RIGHT_PARENTHESIS BITWISE_XOR, REMAINDER COMMA END_OF_INSTRUCTION RETURN, LESS_THAN LAZY_AND, PLUS, LAZY_OR INVERSE_DIVIDE, END, TIMES ELSE_IF, DIVIDE GREATER_OR_EQUALS_THAN

<unaryexpression></unaryexpression>	INTEGER, REAL, BOOLEAN NEGATION, BITWISE_NOT READ_REAL LEFT_PARENTHESIS IDENTIFIER, BOOLEAN_CAST PRINTLN, READ_INTEGER PLUS, MINUS, INTEGER_CAST REAL_CAST	ARITHMETIC_SHIFT_RIGHT EQUALITY, BITWISE_AND ARITHMETIC_SHIFT_LEFT TERNARY_IF, GREATER_THAN LESS_OR_EQUALS_THAN ELSE, TERNARY_ELSE, POWER INEQUALITY, BITWISE_OR MINUS, RIGHT_PARENTHESIS BITWISE_XOR, REMAINDER COMMA END_OF_INSTRUCTION RETURN, LESS_THAN LAZY_AND, PLUS, LAZY_OR INVERSE_DIVIDE, END, TIMES ELSE_IF, DIVIDE GREATER_OR_EQUALS_THAN
$<\!$	READ_INTEGER, INTEGER REAL, PLUS, BOOLEAN, MINUS BITWISE_NOT, READ_REAL INTEGER_CAST LEFT_PARENTHESIS BOOLEAN_CAST, IDENTIFIER PRINTLN, REAL_CAST	ARITHMETIC_SHIFT_RIGHT EQUALITY, BITWISE_AND ARITHMETIC_SHIFT_LEFT TERNARY_IF, GREATER_THAN LESS_OR_EQUALS_THAN ELSE, TERNARY_ELSE, POWER INEQUALITY, BITWISE_OR MINUS, RIGHT_PARENTHESIS BITWISE_XOR, REMAINDER COMMA END_OF_INSTRUCTION RETURN, LESS_THAN LAZY_AND, PLUS, LAZY_OR INVERSE_DIVIDE, END, TIMES ELSE_IF, DIVIDE GREATER OR EQUALS THAN
<unaryminusplusexpression></unaryminusplusexpression>	READ_INTEGER, INTEGER REAL, PLUS, BOOLEAN, MINUS READ_REAL, INTEGER_CAST LEFT_PARENTHESIS BOOLEAN_CAST, IDENTIFIER PRINTLN, REAL_CAST	ARITHMETIC_SHIFT_RIGHT EQUALITY, BITWISE_AND ARITHMETIC_SHIFT_LEFT TERNARY_IF, GREATER_THAN LESS_OR_EQUALS_THAN ELSE, TERNARY_ELSE, POWER INEQUALITY, BITWISE_OR MINUS, RIGHT_PARENTHESIS BITWISE_XOR, REMAINDER COMMA END_OF_INSTRUCTION RETURN, LESS_THAN LAZY_AND, PLUS, LAZY_OR INVERSE_DIVIDE, END, TIMES ELSE_IF, DIVIDE GREATER_OR_EQUALS_THAN

<Unary $f Atomic Expression>$	READ_INTEGER, INTEGER REAL, BOOLEAN, READ_REAL INTEGER_CAST LEFT_PARENTHESIS BOOLEAN_CAST, IDENTIFIER PRINTLN, REAL_CAST	ARITHMETIC_SHIFT_RIGHT EQUALITY, BITWISE_AND ARITHMETIC_SHIFT_LEFT TERNARY_IF, GREATER_THAN LESS_OR_EQUALS_THAN ELSE, TERNARY_ELSE, POWER INEQUALITY, BITWISE_OR MINUS, RIGHT_PARENTHESIS BITWISE_XOR, REMAINDER COMMA END_OF_INSTRUCTION RETURN, LESS_THAN LAZY_AND, PLUS, LAZY_OR INVERSE_DIVIDE, END, TIMES ELSE_IF, DIVIDE GREATER_OR_EQUALS_THAN
<binaryexpression></binaryexpression>	READ_INTEGER, INTEGER REAL, PLUS, BOOLEAN NEGATION, MINUS BITWISE_NOT, READ_REAL INTEGER_CAST BOOLEAN_CAST LEFT_PARENTHESIS IDENTIFIER, PRINTLN REAL_CAST	TERNARY_IF RIGHT_PARENTHESIS, END COMMA END_OF_INSTRUCTION ELSE_IF, ELSE TERNARY_ELSE, RETURN
<binaryexpression'></binaryexpression'>	LAZY_OR, EPSILON_VALUE	TERNARY_IF RIGHT_PARENTHESIS, END COMMA END_OF_INSTRUCTION ELSE_IF, ELSE TERNARY ELSE, RETURN
<binarylazyorexpression></binarylazyorexpression>	READ_INTEGER, INTEGER REAL, PLUS, BOOLEAN NEGATION, MINUS BITWISE_NOT, READ_REAL INTEGER_CAST LEFT_PARENTHESIS BOOLEAN_CAST, IDENTIFIER PRINTLN, REAL_CAST	TERNARY_IF, LAZY_OR RIGHT_PARENTHESIS, END COMMA END_OF_INSTRUCTION ELSE_IF, ELSE TERNARY_ELSE, RETURN
<binarylazyorexpression'></binarylazyorexpression'>	LAZY_AND, EPSILON_VALUE	TERNARY_IF, LAZY_OR RIGHT_PARENTHESIS, END COMMA END_OF_INSTRUCTION ELSE_IF, ELSE TERNARY_ELSE, RETURN
$<\!$	READ_INTEGER, INTEGER REAL, PLUS, BOOLEAN NEGATION, MINUS BITWISE_NOT, READ_REAL INTEGER_CAST BOOLEAN_CAST LEFT_PARENTHESIS IDENTIFIER, PRINTLN REAL_CAST	LAZY_AND, TERNARY_IF LAZY_OR RIGHT_PARENTHESIS, END COMMA END_OF_INSTRUCTION ELSE_IF, ELSE TERNARY_ELSE, RETURN

$<\!$	EQUALITY, INEQUALITY GREATER_THAN EPSILON_VALUE LESS_OR_EQUALS_THAN GREATER_OR_EQUALS_THAN LESS_THAN	LAZY_AND, TERNARY_IF LAZY_OR RIGHT_PARENTHESIS, END COMMA END_OF_INSTRUCTION ELSE_IF, ELSE TERNARY_ELSE, RETURN
<binarynumericexpression></binarynumericexpression>	READ_INTEGER, INTEGER REAL, PLUS, BOOLEAN NEGATION, MINUS BITWISE_NOT, READ_REAL INTEGER_CAST LEFT_PARENTHESIS BOOLEAN_CAST, IDENTIFIER PRINTLN, REAL_CAST	EQUALITY, TERNARY_IF GREATER_THAN RIGHT_PARENTHESIS, COMMA END_OF_INSTRUCTION LESS_OR_EQUALS_THAN ELSE, TERNARY_ELSE, RETURN LESS_THAN, LAZY_AND INEQUALITY, LAZY_OR, END ELSE_IF GREATER_OR_EQUALS_THAN
<binarynumericexpression'></binarynumericexpression'>	BITWISE_OR, PLUS, MINUS BITWISE_XOR EPSILON_VALUE	EQUALITY, TERNARY_IF GREATER_THAN RIGHT_PARENTHESIS, COMMA END_OF_INSTRUCTION LESS_OR_EQUALS_THAN ELSE, TERNARY_ELSE, RETURN LESS_THAN, LAZY_AND INEQUALITY, LAZY_OR, END ELSE_IF GREATER_OR_EQUALS_THAN
<binarytermexpression></binarytermexpression>	READ_INTEGER, INTEGER REAL, PLUS, BOOLEAN NEGATION, MINUS BITWISE_NOT, READ_REAL INTEGER_CAST BOOLEAN_CAST LEFT_PARENTHESIS IDENTIFIER, PRINTLN REAL_CAST	EQUALITY, TERNARY_IF RIGHT_PARENTHESIS GREATER_THAN BITWISE_XOR, COMMA END_OF_INSTRUCTION LESS_OR_EQUALS_THAN ELSE, TERNARY_ELSE, RETURN LESS_THAN, LAZY_AND INEQUALITY, BITWISE_OR PLUS, LAZY_OR, MINUS, END ELSE_IF GREATER_OR_EQUALS_THAN
<binarytermexpression'></binarytermexpression'>	ARITHMETIC_SHIFT_RIGHT ARITHMETIC_SHIFT_LEFT EPSILON_VALUE	EQUALITY, TERNARY_IF RIGHT_PARENTHESIS GREATER_THAN BITWISE_XOR, COMMA END_OF_INSTRUCTION LESS_OR_EQUALS_THAN ELSE, TERNARY_ELSE, RETURN LESS_THAN, LAZY_AND INEQUALITY, BITWISE_OR PLUS, LAZY_OR, MINUS, END ELSE_IF GREATER_OR_EQUALS_THAN

		ARITHMETIC SHIFT RIGHT
<binaryshiftedexpression></binaryshiftedexpression>	READ_INTEGER, INTEGER REAL, PLUS, BOOLEAN NEGATION, MINUS BITWISE_NOT, READ_REAL INTEGER_CAST LEFT_PARENTHESIS BOOLEAN_CAST, IDENTIFIER PRINTLN, REAL_CAST	EQUALITY, TERNARY_IF ARITHMETIC_SHIFT_LEFT RIGHT_PARENTHESIS GREATER_THAN BITWISE_XOR, COMMA END_OF_INSTRUCTION LESS_OR_EQUALS_THAN ELSE, TERNARY_ELSE, RETURN LESS_THAN, LAZY_AND INEQUALITY, PLUS BITWISE_OR, LAZY_OR, MINUS END, ELSE_IF GREATER_OR_EQUALS_THAN
$<\!$	BITWISE_AND INVERSE_DIVIDE REMAINDER, TIMES EPSILON_VALUE, DIVIDE	ARITHMETIC_SHIFT_RIGHT EQUALITY, TERNARY_IF ARITHMETIC_SHIFT_LEFT RIGHT_PARENTHESIS GREATER_THAN BITWISE_XOR, COMMA END_OF_INSTRUCTION LESS_OR_EQUALS_THAN ELSE, TERNARY_ELSE, RETURN LESS_THAN, LAZY_AND INEQUALITY, PLUS BITWISE_OR, LAZY_OR, MINUS END, ELSE_IF GREATER_OR_EQUALS_THAN
<binaryfactorexpression></binaryfactorexpression>	READ_INTEGER, INTEGER REAL, PLUS, BOOLEAN NEGATION, MINUS BITWISE_NOT, READ_REAL INTEGER_CAST BOOLEAN_CAST LEFT_PARENTHESIS IDENTIFIER, PRINTLN REAL_CAST	ARITHMETIC_SHIFT_RIGHT EQUALITY, BITWISE_AND ARITHMETIC_SHIFT_LEFT TERNARY_IF, GREATER_THAN LESS_OR_EQUALS_THAN ELSE, TERNARY_ELSE INEQUALITY, BITWISE_OR MINUS, RIGHT_PARENTHESIS BITWISE_XOR, REMAINDER COMMA END_OF_INSTRUCTION RETURN, LESS_THAN LAZY_AND, PLUS, LAZY_OR INVERSE_DIVIDE, END, TIMES ELSE_IF, DIVIDE GREATER_OR_EQUALS_THAN

$<\!$	POWER, EPSILON_VALUE	ARITHMETIC_SHIFT_RIGHT EQUALITY, BITWISE_AND ARITHMETIC_SHIFT_LEFT TERNARY_IF, GREATER_THAN LESS_OR_EQUALS_THAN ELSE, TERNARY_ELSE INEQUALITY, BITWISE_OR MINUS, RIGHT_PARENTHESIS BITWISE_XOR, REMAINDER COMMA END_OF_INSTRUCTION RETURN, LESS_THAN LAZY_AND, PLUS, LAZY_OR INVERSE_DIVIDE, END, TIMES ELSE_IF, DIVIDE GREATER_OR_EQUALS_THAN
<if></if>	IF	END, END_OF_INSTRUCTION ELSE IF, ELSE, RETURN
<ifend></ifend>	END, ELSE_IF, ELSE	END, END_OF_INSTRUCTION ELSE_IF, ELSE, RETURN
$<\!$	READ_INTEGER, READ_REAL INTEGER_CAST BOOLEAN_CAST, PRINTLN REAL_CAST	ARITHMETIC_SHIFT_RIGHT EQUALITY, BITWISE_AND TERNARY_IF ARITHMETIC_SHIFT_LEFT GREATER_THAN LESS_OR_EQUALS_THAN TERNARY_ELSE, ELSE, POWER INEQUALITY, BITWISE_OR MINUS, RIGHT_PARENTHESIS REMAINDER, BITWISE_XOR COMMA END_OF_INSTRUCTION RETURN, LESS_THAN LAZY_AND, PLUS, LAZY_OR INVERSE_DIVIDE, TIMES, END ELSE_IF, DIVIDE GREATER_OR_EQUALS_THAN
<functioncalltail></functioncalltail>	LEFT_PARENTHESIS	ARITHMETIC_SHIFT_RIGHT EQUALITY, BITWISE_AND TERNARY_IF ARITHMETIC_SHIFT_LEFT GREATER_THAN LESS_OR_EQUALS_THAN TERNARY_ELSE, ELSE, POWER INEQUALITY, BITWISE_OR MINUS, RIGHT_PARENTHESIS REMAINDER, BITWISE_XOR COMMA END_OF_INSTRUCTION RETURN, LESS_THAN LAZY_AND, PLUS, LAZY_OR INVERSE_DIVIDE, TIMES, END ELSE_IF, DIVIDE GREATER_OR_EQUALS_THAN

<parameter></parameter>	INTEGER, REAL, BOOLEAN NEGATION, BITWISE_NOT READ_REAL, EPSILON_VALUE BOOLEAN_CAST, IDENTIFIER LEFT_PARENTHESIS PRINTLN, READ_INTEGER PLUS, MINUS, INTEGER_CAST REAL_CAST	RIGHT_PARENTHESIS
<parametertail></parametertail>	COMMA, EPSILON_VALUE	RIGHT_PARENTHESIS
<functiondefinition></functiondefinition>	FUNCTION	END, END_OF_INSTRUCTION ELSE_IF, ELSE, RETURN
<functiondefinitionend></functiondefinitionend>	END, RETURN	END, END_OF_INSTRUCTION ELSE_IF, ELSE, RETURN
<argument></argument>	EPSILON_VALUE IDENTIFIER	RIGHT_PARENTHESIS
<argumenttail></argumenttail>	COMMA, EPSILON_VALUE	RIGHT_PARENTHESIS

# 4 Action Table

RETURN			6	11								30			90	60				51		54	62		89	7.5	1	62	60	5						66		
FUNCTION	-	7	$\infty$		Ţ	Ţ				T		I			T				$\prod$			I						Ţ	$\Box$						86		J	_
PRINTLN	Н	_	$\infty$		_	_					28	ì		36		41	3 43 43 43 43 43	46	47	45	52 52 52 52	1	66 66 66 66	63 63 63 63 63 63	02 02 02	80	73 73 73 73 73	0	080080080	$\perp$		92	2	24			$\perp$	_
BOOLEAN_CAST	-		$\infty$		_	4	_	_		_	28	i L		936	4	41	843	94	47	45	252	1	<u> </u>	963	00	3	73	- 6	<u>8</u>	$\perp$		191	- 2	74	_	Ш	$\dashv$	_
REAL_CAST INTEGER CAST	-	_	$\infty$		_	_		-		_	28 28 28 28 28	í		3636363	_	-41	45	<u>¥</u>	47	745	255	7	2	365	9	2	75	0	<u>×</u>	+		88 89 90 91	è	94 94 94	-		$\dashv$	_
DEAD INTEGER	F	_	×		$\dashv$	+	-	-		+	328	í		33	+	141	345	346	4.7	45	255	7	<u>~</u>	365	- 0	<u> </u>	37.	- 6	<u>×</u>	+		388	- 6	8	-	$\Box$	$\dashv$	_
READ_INTEGER READ_REAL	П		$\infty$		_					4	25	í		8		141	45	94 :	47	45	255	7	2	39	Ü	<u> </u>	22			+			è	200			$\dashv$	_
INTEGER TYPE	г	9	$\infty$		_	_	-	-				í		36	_	4	43	4;	4	7	52	į	င္ပင္	99	- 0	ő	73,	- 2	20	+		82	è	94	-	Н	$\dashv$	_
REAL TYPE	⊢	-	╄		-	+	_	_		2	4	_		-	_		Н	-	+		Н		+	-	+	_	+	_	_	+			_	+	-	Н	+	_
BOOLEAN TYPE	┡		╀		$\dashv$	+	+	-		90	1	+		$\dashv$	+		$\vdash$	$\dashv$	+		Н		+	$\vdash$	+	+		_	_	+		$\dashv$	_	+	+	$\vdash$	$\dashv$	_
	┝	ļ.,	ļ.		$\dashv$	+	-	-		C TT	ĭ			$\dashv$	4		$\vdash$	4	+		Н		_	-	+	+				+				+	-	Н	$\dashv$	_
LET	F	4			_			19		-		_		_	_		$\perp$	_	4					-	~		1			a	,0		_	_			$\dashv$	
ELSE	_			11	_	_				_		30		_	06	ő		_	4	51	Ш	154	62		898	7.0	-	62	60	5	185			$\perp$			$\dashv$	_
ELSE_IF	_		6		_	_				_		30		_	06	ő		_	_	51	Н	54	62	-	89	7.5	-	79	ŏ	70 ~	84			$\perp$	-		$\dashv$	_
IF	-	5	$\infty$		$\rightarrow$	$\perp$	_	_	20	4		_		$\dashv$		$\perp$	$\sqcup$	4	$\perp$		Ш			_	$\perp$	_		_	_	83		$\Box$		+	_	Ш	$\dashv$	_
POWER	_				$\dashv$	4				4				_	06 06 06	ő	$\perp$	4	$\perp$		Ш		_	_	_	4		~	0	5				$\perp$	-		$\dashv$	_
INVERSE_DIVIDE															6	30	$\perp$											28	S								_	_
BITWISE_AND	_													_	- 6	~	$\sqcup$						$\perp$					77	60	5							$\dashv$	_
REMAINDER			╙		_	_				_				_	- 6	ñ	Ш	_	$\perp$		Ш				_	$\perp$		92	S	5				$\perp$		Ш	_	
DIVIDE			╙		_	4				_		_		_	- 6	ñ	Ш	_	$\perp$		Ш				_	$\perp$		72	ç	2				$\perp$	_	Ш	_	_
TIMES															90	<u>ه</u>												74	0000	2								
ARITHMETIC_SHIFT_RIGHT															06 06 06 06 06	, e	Ш									7	-	79	S	07							$\perp$	
ARITHMETIC_SHIFT_LEFT															90	S C					Ш					70	2	73	60 60	5								
BITWISE_XOR				$\square$									$\square$		9	6 <u> </u>					$\square$				67	7.0	1	29	S	70						$\Box$	$\Box$	_
BITWISE_OR	L	L	$\perp$	$\Box$	[	$\Box$	╧	$\perp$	$\Box \Gamma$	⅃	$\perp$	$\perp$	$\Box$	J	90		ĹĬ	T	$\Box$	$\perp$	$\Box$	_[	$\bot$	$\perp$	99	7.	1	<u>1</u> 20	ွ	20		ĹĴ	$\Box$	$\bot$	$\perp$	ĹĴ	[	_
INEQUALITY	L	Γ	$\Box$		T		$oxed{oxed}$			⅃	$\Box$			$_{\perp}$ T	00		ĹŢ	$\Box$		$oldsymbol{oldsymbol{oldsymbol{oldsymbol{I}}}$	┰┚	$\Box$ $\Box$	61	L	89	7.5	1	79	0	7	П	LT		$oxed{\Box}$		LΤ	$_{\perp}$ T	_
EQUALITY										T		Ι			06 06 06 06	60							09	$\Gamma$	89	15	1	62	69 69 69	70								_
LESS_OR_EQUALS_THAN										T				T	000	60	П						596061		29 99 89 89 89 89 89 89 89 89 89 89 89 89	62 62 62 62 62	1	79 79 79 79 79 79 79 79 79 79 79 74	18	3								
GREATER_OR_EQUALS_THAN															06 06 06 06	60	П						25		89	13	1	79	ြင္လ	2							$\Box$	_
LESS_THAN															00	80			$\neg$				22		8	2	1	29	69 69 69	7							Т	
GREATER THAN					$\neg$	$\top$				$\top$		$\top$		$\neg$		6			$\top$		П		190		8	13	1	62	C C	4		П		$\top$		П	$\top$	_
LAZY AND					$\dashv$	$\top$								$\neg$	2	2			$\top$		П	53	32		8	3	1	62	Š	7				$\top$			$\top$	_
LAZYOR	$\vdash$		T		$\dashv$	$\top$				$\top$				$\dashv$	06	2		1	$\top$	20	П	54	62625657		8	62 62 62 62	1	62	S	7				$\top$		П	$\top$	-
RIGHT_PARENTHESIS			T		$\neg$							30		$\neg$	06	2		7	$\top$	51		54	62		89	3		. 62	ç				L	26.			102	4
	_		-							_	~			_	٥	٠ <u> </u>			~	5			9	_	9	1	-		_ 0	٥					)		=	$\equiv$
LEFT_PARENTHESIS	_				-	15				_	3 28	í		_	96	09 09 00 1	4	4	4 5	#	52	ì	3	99	1 0	ő	73 73 73 73 73 73 73 73 73	_ [	$ \bar{x} \mid \bar{x} $	a .			93	94			4	
PLUS	_		_		_	_				_	28	í		_	90	<u>4</u>	45	4	- 5	4	52 52 52 52	7	ee ee ee ee	9	9	3 6	1 22	32	<u>∞</u> ∞	5			2	94			$\dashv$	_
MINUS	_		_		_	_				_	28	í		_	- 6	<u>4</u>	43	4	1	3	25	1	3	63	30 0	3 6	7 62	22	<u> </u>	5			2	22	_	Ш	$\dashv$	_
BITWISE_NOT			╙		_	4				_	2828	i L		_	_	41	42	_	- 5	24	52	1	2	63	9	2	73	0	<u></u>	$\perp$			Č	9.4	_	Ш	_	_
NEGATION										_	20	ì				40			- 5	4. 20	52	1	2	63	0	2	73	0	$\frac{2}{\infty}$	$\perp$			2	94			$\perp$	
BOOLEAN											28	ì		35		41	43 43	46	47	43	52 52	7	ဂ္ဂ	63	0	80	73	C	$\frac{2}{\infty}$				2	94				
REAL											28	ì		34		41	43	46	47	45	52	) )	၁၁	63	0	60	73	8	<u></u>	$\perp$			2	94				
INTEGER											28	ì		33		41	43	46	47	43	52	1	ဂ္ဂ	63	0	80	73	0	<u>∞</u>				2	24				_
TERNARY_IF												29			00	60				51		54	- 19		9	4	-	<u>-1</u>	0	١ ٥								
COMMA					9	13	-					30			90	93				51		54	62		89	7.0	1	79	60	4				96			9	103
TERNARY ELSE	$\vdash$	+	+			-	+	+	-	77	-	0	31	$\dashv$	06	מ	+	$\dashv$	+	51		54	62		89	7.0	1	79	60	1				+	+	Н	+	=
FOR		,_	$\infty$		$\dashv$	+	+	$\vdash$	22	71		<u> </u>	3	$\dashv$	- 6	<u>-                                    </u>	+	$\dashv$	+	5	+	ಬ	-19	+	9	1	-		- 0	+		$\dashv$		+	+	$\vdash$	+	—
WHILE	Ε.	20	~		$\dashv$	+			212	+				$\dashv$	+		+	-	+		+		+	+	+	+				+				+			+	—
INTEGER_TYPE	Ε.	2.7	+~		$\dashv$	+	+	$\vdash$	2	1	+	+		$\dashv$	+		+	-	+		+		-	+	+	+			+	+			-	+		Н	+	—
REAL TYPE	$\vdash$		$\vdash$		+	+				<u>ت</u>	7			$\dashv$	+		$\vdash$	-	+		Н		+	+	+	+				+		$\vdash$		+	-	$\vdash$	+	_
BOOLEAN TYPE	$\vdash$	-	+		+	+	+	$\vdash$	+	96 96	4	+		$\dashv$	+	+	$\vdash$	+	+	+	$\vdash$	$\dashv$	+	$\vdash$	+	+	+	+	+	+	$\vdash$	$\vdash$	+	+	+	$\vdash$	+	_
END OF INSTRUCTION	$\vdash$	-	-		+	+	+	+	-		4	-		$\dashv$		+	$\vdash$	+	+	-	$\vdash$	₩.	-	$\vdash$	<u>~</u>		1	6	4	1	$\vdash$	$\vdash$	+	+	+	$\vdash$	+	_
	F	-	10	10	٠,	20 10		-		72	-	30		$\dashv$	9	99	$\vdash$	$\dashv$	+	51	$\vdash$	54	62	$\perp$	89	7.5	-	79	CO	<u> </u>		$\vdash$	$\perp$	+	+	$\vdash$	$\dashv$	_
ASSIGNATION		-	-		-  ;	13	1	-	$\perp$	$\perp$	$\bot$	+		_	4	_	$\vdash$	4	$\perp$	$\bot$	Н	_	$\perp$	$\perp$	4	$\perp$	$\perp$	_	$\downarrow$	_	$\vdash$	$\square$	4	$\perp$	+		$\dashv$	_
END			6	Ξ								30			00	99				51		54	62		89	3	1	79	S	5	98					100		
CONST	П	3	$\infty$	П	$\neg$	$\top$	18		$\top$	$\top$	1		П	$\dashv$	$\top$		П	$\dashv$	$\top$	1	П	7	$\top$		$\top$	$\top$			$\top$	$\top$		П	1	$\top$			$\top$	_
TYPE DEFINITION		t				41	Ť		$\top$	$\top$	$\top$			$\dashv$	$\top$	$\top$	Ħ	$\dashv$	$\top$	$\top$	П	$\dashv$	$\top$		$\top$	$\top$	$\top$		$\top$	$\top$		H	1	$\top$		П	+	_
IDENTIFIER		2	$\infty$	$\Box$	12	+	+		+	+	28	1		32	<u>,                                    </u>	-	43	46	<u> </u>	649	25	ı	22	63		60	73		2	+		$\vdash$	_	75	+	$\vdash$	<u></u> =†	_
	F	.4	1~		-	$\perp$	-		$\perp$	$\perp$	2	_		<u>د</u>	_	41	4	4	47	_	+	_	_	_	-	_	-	_	_	$\perp$		Щ			-	Ш	102 101	_
EPSILON_VALUE	-		6	Ξ								30			00	ဂိ				51		54	62		88	7.5	-	79	S	5			Č	6.			201	104
	T		T		7	1								$\exists$	_				$\top$		П				T	$\top$				$\top$				$\top$			Ť	_
						,							١. ا			a		4	Λ l			$\wedge$	<u> </u>	A	Λ	۱,		Λ	۸ ا	\								
					A :	<u></u>						$\wedge$			ioi.	∄	Sio	. <u>[</u> ]	äΙ		OH OH	'n.	<u> </u>	ion	U C	<u>ا</u> اد	][8]	,ŭ	äľ,	ì		$\wedge$			_	٩l		
			$\setminus$	E	E E	<u>::</u> ]]	Δ					on	sio	싊	es   :		les	şë	SSI		SSi	SSi	ess Sej	38S	SSi		SSi	·SSi	SSI   SSI	210		all	4		Ιġ	찝		Λ
	1	7	ist.	ŧΤε	ĘĘ.	01	ion			, [	^	SSi	res	Sic	ĬĮ.	3i   E	X	Хр	ig	Sic	pre	ore	텕	pr.	pre	<u> </u>	ore [5	ore	re E	<u> </u>		입	Tail	جَ ا رُ	iţi	on	V E	3il
	E E	ior	la I	Zis.	ĮĮ.	al c	nit.				٥. ا	pre	χĎ	Si	원]	š Š	围	떫.	Χį	ies   ies	X	뛏	퉑	EX	ÄΠ	취	웬և	揻.	3	쉰.	$\Delta$	tio	all	ī] šē	цij	liti	ent	ij
	gra	uct	tio	Juc	si.	il št		<block></block>	< Toop	< FOT Lall	\tau ype \tag{xpressio}	K	eE.	dy.	fiel F	ŽĮ,	[일]	إ إثر	틸		딘	뙨		ric	G [6			闎	뷥녆		<IfEnd $>$	nc	입	휘	De	빏	ğΓ	<argument tail<="" td=""></argument>
	rog	stri	n.	3tic		<u>i</u>		B	IJ,	Ē [f		則	Els	띵	nti.	围	ise	ISF	틹	ŢĺĘ		7 <u>7</u>	Y A	me	nei	#   E		lfte.	£ £		IEE	뢰	Fio.	r   ř	On	اقِا	rgr	un
	< Program	<instruction></instruction>	<instructionlist< p=""></instructionlist<>	<instructionlisttail< p=""></instructionlisttail<>	<identifierinstruction< td=""><td><identification p="" tail<=""></identification></td><td><assignation lany<="" p=""><constdefinition></constdefinition></assignation></td><td>\ \</td><td>\ \ \</td><td>7∣`</td><td>&lt; Type&gt;</td><td><ternaryifexpression></ternaryifexpression></td><td><ternaryelseexpression></ternaryelseexpression></td><td>&lt; Atomic Expression</td><td><pre><atomicidentifierexpression> AtomicIdentiferExpression&gt;</atomicidentifierexpression></pre></td><td><ul> <li>Acouncident in Expression 1an &gt;</li> <li>UnaryExpression&gt;</li> </ul></td><td><unarybitwisenotexpression></unarybitwisenotexpression></td><td><unaryminusplusexpression< td=""><td><unaryatomicexpression></unaryatomicexpression></td><td>&lt; binary expression &lt; Binary Expression'</td><td><binarylazyorexpression></binarylazyorexpression></td><td><binarylazyorexpression'></binarylazyorexpression'></td><td>&lt; BinaryLazyAndExpression &gt; &lt; BinarvLazyAndExpression '&gt;</td><td><binarynumericexpression></binarynumericexpression></td><td><binarynumericexpression< p=""></binarynumericexpression<></td><td></td><td><binaryshiftedexpression></binaryshiftedexpression></td><td><binaryshiftedexpression< td=""><td><pre><binaryfactorexpression> /BinaryFactorExpression&gt;</binaryfactorexpression></pre></td><td>۲  </td><td><math>\vee</math></td><td><builtinfunctioncall></builtinfunctioncall></td><td><functioncalltail< p=""></functioncalltail<></td><td>&lt; Parameter &gt; &lt; Parameter Tail &gt;</td><td><functiondefinition></functiondefinition></td><td><functiondefinitionend></functiondefinitionend></td><td><argument></argument></td><td>rgi</td></binaryshiftedexpression<></td></unaryminusplusexpression<></td></identifierinstruction<>	<identification p="" tail<=""></identification>	<assignation lany<="" p=""><constdefinition></constdefinition></assignation>	\ \	\ \ \	7∣`	< Type>	<ternaryifexpression></ternaryifexpression>	<ternaryelseexpression></ternaryelseexpression>	< Atomic Expression	<pre><atomicidentifierexpression> AtomicIdentiferExpression&gt;</atomicidentifierexpression></pre>	<ul> <li>Acouncident in Expression 1an &gt;</li> <li>UnaryExpression&gt;</li> </ul>	<unarybitwisenotexpression></unarybitwisenotexpression>	<unaryminusplusexpression< td=""><td><unaryatomicexpression></unaryatomicexpression></td><td>&lt; binary expression &lt; Binary Expression'</td><td><binarylazyorexpression></binarylazyorexpression></td><td><binarylazyorexpression'></binarylazyorexpression'></td><td>&lt; BinaryLazyAndExpression &gt; &lt; BinarvLazyAndExpression '&gt;</td><td><binarynumericexpression></binarynumericexpression></td><td><binarynumericexpression< p=""></binarynumericexpression<></td><td></td><td><binaryshiftedexpression></binaryshiftedexpression></td><td><binaryshiftedexpression< td=""><td><pre><binaryfactorexpression> /BinaryFactorExpression&gt;</binaryfactorexpression></pre></td><td>۲  </td><td><math>\vee</math></td><td><builtinfunctioncall></builtinfunctioncall></td><td><functioncalltail< p=""></functioncalltail<></td><td>&lt; Parameter &gt; &lt; Parameter Tail &gt;</td><td><functiondefinition></functiondefinition></td><td><functiondefinitionend></functiondefinitionend></td><td><argument></argument></td><td>rgi</td></binaryshiftedexpression<></td></unaryminusplusexpression<>	<unaryatomicexpression></unaryatomicexpression>	< binary expression < Binary Expression'	<binarylazyorexpression></binarylazyorexpression>	<binarylazyorexpression'></binarylazyorexpression'>	< BinaryLazyAndExpression > < BinarvLazyAndExpression '>	<binarynumericexpression></binarynumericexpression>	<binarynumericexpression< p=""></binarynumericexpression<>		<binaryshiftedexpression></binaryshiftedexpression>	<binaryshiftedexpression< td=""><td><pre><binaryfactorexpression> /BinaryFactorExpression&gt;</binaryfactorexpression></pre></td><td>۲  </td><td><math>\vee</math></td><td><builtinfunctioncall></builtinfunctioncall></td><td><functioncalltail< p=""></functioncalltail<></td><td>&lt; Parameter &gt; &lt; Parameter Tail &gt;</td><td><functiondefinition></functiondefinition></td><td><functiondefinitionend></functiondefinitionend></td><td><argument></argument></td><td>rgi</td></binaryshiftedexpression<>	<pre><binaryfactorexpression> /BinaryFactorExpression&gt;</binaryfactorexpression></pre>	۲	$\vee$	<builtinfunctioncall></builtinfunctioncall>	<functioncalltail< p=""></functioncalltail<>	< Parameter > < Parameter Tail >	<functiondefinition></functiondefinition>	<functiondefinitionend></functiondefinitionend>	<argument></argument>	rgi
	`	V	H	nst	<u>e</u> ]	Ħ	ğ  S				\	<u>  Ĭ</u>	na	150			l <u>iā</u> ļ	Ϋ́		"in	ry	Z	<u> </u>	Ţ		1   1		Ty,	ary	<u> </u>		1	弫[	٧ <del>[ ]</del>	Ę	]ct	V -	$\stackrel{\wedge}{A}$
			\	1	뭐.	ا <u>ا</u> ق	/					Ĕ	<u>l'er</u>	$\vec{\lor}$	m l	3 V	ary	ar	lna L	715	lina	na	nan Jar	na	naı	<u> </u>	ii.	ina	ğ .	Ĭ		Ψĺ	V	\	7	E.		٧
					V	₹L						1	V	- [.	<u> </u>	5	lğ,	되	$\mathbb{C}$		m	Bi	캙	ΒÏ	Bi:	715	1191	m f	4 J	٦.		V			1	$\nabla$		
																														/							- 1	
															V 2	4	\forall	abla	`			Λ.	$\nabla  \nabla$	V	$\nabla$	`   `	\	$\vee$	` `									