Grammar	First	Follow
1- program -> stmt_seq	If, id, int, float, char	$
2- stmt_seq -> stmt stmt_seq'	If, id, int, float, char	$
3- stmt_seq' -> stmt_seq | 3	if, id, int, float, char, 3	$, }
4- stmt -> if_stmt | assign_stmt ; | declare_stmt ;	If, id, int, float, char	if, id, int, float, char, $
5- if_stmt -> if (condition) { stmt_seq' } else_part	If	if, id, int, float, char, $
6- else_part -> else { stmt_seq' } | 3	else, 3	if, id, int, float, char, $
7- condition -> exp condition'	(, number, id	)
8- condition' -> comp_sign exp | 3	<, >, ==, >=, <=, 3	)
9- comp_sign -> < | > | == | >= | <=	<, >, ==, >=, <=	(, number, id
10- exp -> term exp'	(, number, id	<, >, ==, >=, <=,), ;

`11- exp' -> add_op term exp' | 3`	+, -, 3	<, >, ==, >=, <=,), ;
`12- add_op -> + | -`	+, -	(, number, id
`13- term -> factor term'`	(, number, id	+, -, <, >, ==, >=, <=,), ;
`14- term' -> mul_op factor term' | 3`	*, /, 3	+, -, <, >, ==, >=, <=,), ;
`15- mul_op -> * | /`	*, /	(, number, id
`16- factor -> ( exp ) | number | id`	(, number, id	*, /, +, -, <, >, ==, >=, <=,), ;
`17- declare_stmt -> datatype id x`	Int, float, char	;
`18- x -> = exp | 3`	=, 3	;
`19- assign_stmt -> id = exp`	Id	;
`20- datatype-> int | float | char`	Int, float, char	id

Terminals = if, id, int, float, char, else, (,), <, >, =, <=, >=, ==, number, *, /, +, {, }, ;, $

If id int float char else ( ) < > = <= >= == ; num * / + - } $

LL1 parser table

	If	Id	Int	Float	Char	Else	(	)	<	>	=	<=	>=	==	;	Num	*	/	+	-	}	$
1-	1	1	1	1	1																	
2-	2	2	2	2	2																	
3-	Stmt_seq’->stmt_seq	Stmt_seq’->stmt_seq	Stmt_seq’->stmt_seq	Stmt_seq’->stmt_seq	Stmt_seq’->stmt_seq																Stmt_seq’->3	Stmt_seq’->3
4-	Stmt->if_stmt	stmt->assign_stmt	Stmt->declare_stmt	Stmt->declare_stmt	Stmt->declare_stmt																	
5-	5																					

	If	id	int	float	char	else	(	)	<	>	=	<=	>=	==	;	num	*	/	+	-	}	$
6-	else_part->3	else_part->3	else_part->3	else_part->3	else_part->3	else_part->else{ stmt_seq’ }																else_part->3
7-		7					7									7						
8-								Condition’->3	condition’->comp_sign exp	condition’->comp_sign exp	condition’->comp_sign exp	condition’->comp_sign exp	condition’->comp_sign exp	condition’->comp_sign exp								
9-									comp_sign-> <	comp_sign-> >	comp_sign-> =	comp_sign-> <=	comp_sign-> >=	comp_sign-> ==								
10-		10					10									10						
11-								exp’-> 3	exp’-> 3	exp’-> 3	exp’-> 3	exp’-> 3	exp’-> 3	exp’-> 3	exp’-> 3				exp’->add_op term exp’	exp’->add_op term exp’		
12-																			add_op-> +	add_op-> -		

	If	id	int	float	char	else	(	)	<	>	=	<=	>=	==	;	num	*	/	+	-	}	$
13-		13					13									13						
14-									Term’->3	Term’->3	Term’->3	Term’->3	Term’->3	Term’->3	Term’->3		Term->mul_op factor term’	Term->mul_op factor term’	Term’->3	Term’->3		
15-																	mul_op->*	mul_op->/				
16-		Factor->id					Factor->(exp)									Factor->num						
17-			17	17	17																	
18-											x-> = exp				x-> 3							
19-		19																				
20			datatype->int	Datatype->float	datatype->char																	