No. of non-terminals = 45					
No. of productions = 88					
Comments	**RULE**	**Productions**	**FIRST**	**FOLLOW**	
<h> is used to eliminate the ambiguity as both <data decls> and <func list> start with "<type name> ID" and both have an ebsilon present in the rules as well. It works in a way like **<program> => <data decls><func list> | <data decls> | <func list> | e** <i> is used to implement **<program> => <data decls><func list> | <data decls>** part of the above statement	<Goal>	<program>	{int, void, binary, decimal,empty}	{$}	
	<program>	<type name> ID <h> | **empty**	{int, void, binary, decimal,empty}	{$}	
	<h>	<f> <id list prime> **semicolon** <data decls> <i> | **left_parenthesis** <parameter list> **right_parenthesis** <g> <func list>	{left_bracket, left_parenthesis, empty}	{$}	
	<i>	<func list> | **empty**	{int, void, binary, decimal,empty}	{$}	
	<func list>	<func><func list> | **empty**	{int, void, binary, decimal,empty}	{$}	
<g> is used as <func decl> is common to the two productions for Rule <func>	<func>	<func decl> <g>	{int, void, binary, decimal}	{int, void, binary, decimal, $}	
	<g>	**semicolon** | **left_brace** <data decls> <statements> **right_brace**	{semicolon, left_brace}	{int, void, binary, decimal, $}	
	<func decl>	<type name> **ID** (<parameter list>)	{int, void, binary, decimal}	{semicolon, left_brace}	
	<type name>	**int** | **void** | **binary** | **decimal**	{int, void, binary, decimal}	{ID}	
	<parameter list>	**void** | <non-empty list> | **empty**	{int, void, binary, decimal,empty}	{right_parenthesis}	
	<non-empty list>	<type name> **ID** <non-empty list prime>	{int, void, binary, decimal}	{right_parenthesis}	
	<non-empty list prime>	**comma** <type name> **ID** <non-empty list prime> | **empty**	{comma, empty}	{right_parenthesis}	
	<data decls>	<type name> <id list> **semicolon** <data decls> | **empty**	{int, void, binary, decimal,empty}	{int, void, binary, decimal, ID, if, while, return, break, continue, read, write, print, right_brace, $}	
	<id list>	<id> <id list prime>	ID	{semicolon}	
	<id list prime>	**comma** <id> <id list prime> | **empty**	{comma, empty}	{semicolon}	
<f> is used as **ID** is common to the two productions for Rule <id>	<id>	**ID** <f>	ID	{comma, semicolon, equal_sign}	
	<f>	**left_bracket** <expression> **right_bracket |** **empty**	{left_bracket, empty}	{comma, semicolon, equal_sign}	
	<block statements>	**left_brace** <statements> **right_brace**	left_brace	{ID, if, while, return, break, continue, read, write, print, right_brace}	
	<statements>	<statement> <statements> | **empty**	{ID, if, while, return, break, continue, read, write, print, empty	{right_brace}	

<j> is used as <assignment> and <func call> have **ID** common as their first element	<statement>	**ID** <j> | <if statement> | <while statement> | <return statement> | <break statement> | <continue statement> | **read left_parenthesis ID right_parenthesis semicolon |** **write left_parenthesis** <expression> **right_parenthesis semicolon |** **print left_parenthesis STRING right_parenthesis semicolon**	{ID, if, while, return, break, continue, read, write, print}	{ID, if, while, return, break, continue, read, write, print, right_brace}	
	<j>	<f> **equal_sign** <expression> **semicolon** | **left_parenthesis** <expr list> **right_parenthesis semicolon**	{left_bracket, left_parenthesis, empty}	{ID, if, while, return, break, continue, read, write, print, right_brace}	
	<assignment>	<id> **equal_sign** <expression> **semicolon**	{ID}	{ID, if, while, return, break, continue, read, write, print, right_brace}	
	<func call>	**ID left_parenthesis** <expr list> **right_parenthesis semicolon**	{ID}	{ID, if, while, return, break, continue, read, write, print, right_brace}	
	<expr list>	<non-empty expr list> | **empty**	{ID, NUMBER, minus_sign, left_parenthesis, empty}	{right_parenthesis}	
	<non-empty expr list>	<expression> <non-empty expr list prime>	{ID, NUMBER, minus_sign, left_parenthesis}	{right_parenthesis}	
	<non-empty expr list prime>	**comma** <expression> <non-empty expr list prime> | **empty**	{comma, empty}	{right_parenthesis}	
	<if statement>	**if left_parenthesis** <condition expression> **right_parenthesis** <block statements>	{if}	{ID, if, while, return, break, continue, read, write, print, right_brace}	
<b> is used as <condition> is common to the two productions for Rule <condition expression>	<condition expression>	<condition> <b>	{ID, NUMBER, minus_sign, left_parenthesis}	{right_parenthesis}	
	<b>	<condition op> <condition> | **empty**	{double_and_sign, double_or_sign, empty}	{right_parenthesis}	
	<condition op>	**double_and_sign |** **double_or_sign**	{double_and_sign, double_or_sign}	{ID, NUMBER, minus_sign, left_parenthesis}	
	<condition>	<expression> <comparison op> <expression>	{ID, NUMBER, minus_sign, left_parenthesis}	{double_and_sign, double_or_sign, right_parenthesis}	
	<comparison op>	**== |** **!= |** **> |** **>= |** **< |** **<=**	{==, !=, >, >=, <, <= }	ID, NUMBER, minus_sign, left_parenthesis	
	<while statement>	**while left_parenthesis** <condition expression> **right_parenthesis** <block statements>	{while}	{ID, if, while, return, break, continue, read, write, print, right_brace}	
<e> is used as <return> is common to the two productions for Rule <return statement>	<return statement>	**return** <e>	{return}	{ID, if, while, return, break, continue, read, write, print, right_brace}	
	<e>	<expression> **semicolon** | **semicolon**	{ID, NUMBER, minus_sign, left_parenthesis, semicolon}	{ID, if, while, return, break, continue, read, write, print, right_brace}	

	<break statement>	**break semicolon**	{break}	{ID, if, while, return, break, continue, read, write, print, right_brace}	
	<continue statement>	**continue semicolon**	{continue}	{ID, if, while, return, break, continue, read, write, print, right_brace}	
	<expression>	<term> <expression prime>	{ID, NUMBER, minus_sign, left_parenthesis}	{right_bracket, right_parenthesis, semicolon, comma, ==, !=, >, >=, <, <=, double_and_sign, double_or_sign}	
	<expression prime>	<addop> <term> <expression prime> | **empty**	{plus_sign, minus_sign, empty}	{right_bracket, right_parenthesis, semicolon, comma, ==, !=, >, >=, <, <=, double_and_sign, double_or_sign}	
	<addop>	**plus_sign | minus_sign**	{plus_sign, minus_sign}	{ID, NUMBER, minus_sign, left_parenthesis}	
	<term>	<factor> <term prime>	{ID, NUMBER, minus_sign, left_parenthesis}	{right_bracket, right_parenthesis, semicolon, comma, ==, !=, >, >=, <, <=, double_and_sign, double_or_sign, plus_sign, minus_sign}	
	<term prime>	<mulop> <factor> <term prime> | **empty**	{star_sign, forward_slash, empty}	{right_bracket, right_parenthesis, semicolon, comma, ==, !=, >, >=, <, <=, double_and_sign, double_or_sign, plus_sign, minus_sign}	
	<mulop>	**star_sign | forward_slash**	{star_sign, forward_slash}	{ID, NUMBER, minus_sign, left_parenthesis}	
<a> is used as **ID** is common to the first three productions for Rule <factor>	<factor>	**ID** <a> | **NUMBE**R | **minus_sig**n **NUMB**ER | **left_parenthe**sis <expressio**n> right_parenthesis**	{ID, NUMBER, minus_sign, left_parenthesis}	{right_bracket, right_parenthesis, semicolon, comma, ==, !=, >, >=, <, <=, double_and_sign, double_or_sign, plus_sign, minus_sign, star_sign, forward_slash}	
	<a>	**left_bracket** <expression> **right_bracket** | **left_parenthesi**s <expr list> **right_parenthesis | empty**	{left_bracket, left_parenthesis, empty}	{right_bracket, right_parenthesis, semicolon, comma, ==, !=, >, >=, <, <=, double_and_sign, double_or_sign, plus_sign, minus_sign, star_sign, forward_slash}	