Rule	**First**	**Follow**
Program = decls “EOF”	Int, bool, void, EOF	$
decls = typ “id” decls_prime | epsilon	Int, bool, void, epsilon	EOF
decls_prime = vdecl decls | fdecl decls	Semi, lparen	EOF
Fdecl = “lparen” formals_opt “rparen” “LBRACE” vdecl_list stmt_list “RBRACE”	lapren	Int, bool, void, EOF
formals_opt = formal_list | epsilon	Int, bool, void, epsilon	rparen
formal_list = typ “ID” formal_list_prime	Int, bool, void	rparen
Formal_list_prime = “COMMA” formal_list | epsilon	Comma, epsilon	rparen
typ = “INT” | “BOOL” | “VOID”	Int, bool, void	Id
vdecl_list = vdecl vdecl_list | “epsilon”	Epsilon, semi	RETURN, LBRACE, IF, FOR, WHILE, INTLITERAL, TRUE, FALSE, MINUS, NOT, ID, LPAREN, RBRACE
vdecl = “SEMI”	semi	“INT”, “BOOL”, “VOID”, “EOF”, “SEMI”, “RETURN”, RETURN, “LBRACE”, IF, FOR, WHILE, INTLITERAL, TRUE, FALSE, MINUS, NOT, ID, LPAREN, “RBRACE”
stmt_list = stmt stmt_list | epsilon	ε , “RETURN”, RETURN, “LBRACE”, IF, FOR, WHILE, INTLITERAL, TRUE, FALSE, MINUS, NOT, ID, LPAREN, SEMI	RBRACE
stmt = “RETURN” stmt_prime| expr SEMI |“LBRACE” stmt_list RBRACE|	RETURN, “LBRACE”, IF, FOR, WHILE,	RETURN, LBRACE, IF, FOR, WHILE,

IF LPAREN expr RPAREN stmt stmt_prime_prime| FOR LPAREN expr_opt SEMI expr SEMI expr_opt RPAREN stmt| WHILE LPAREN expr RPAREN stmt	INTLITERAL, TRUE, FALSE, MINUS, NOT, ID, LPAREN, SEMI	INTLITERAL, TRUE, FALSE, MINUS, NOT, ID, LPAREN, RBRACE, NOELSE, ELSE, RBRACE
stmt_prime ->SEMI| expr SEMI	Semi, INTLiteral, true, false, id, minus, not, lparen	RETURN, LBRACE, IF, FOR, WHILE, INTLITERAL, TRUE, FALSE, MINUS, NOT, ID, LPAREN, SEMI, RBRACE, NOELSE, ELSE
Stmt_prime_prime = NOELSE | ELSE stmt	NOELSE, ELse	RETURN, “LBRACE”, IF, FOR, WHILE, INTLITERAL, TRUE, FALSE, MINUS, NOT, ID, LPAREN, “RBRACE”, NOELSE, ELSE, RBRACE
Expr_opt = expr | “epsilon”	INTLiteral, true, false, id, minus, not, lparen, epsilon	Semi, rparen
Expr = expr expr_prime_prime | ID expr_prime | LITERAL | TRUE | FALSE | MINUS expr %prec NEG | NOT expr | LPAREN expr RPAREN	INTLiteral, true, false, id, minus, not, lparen	SEMI, RPAREN, PLUS, MINUS, TIMES, DIVIDE, EQ, NEQ, LT, LEQ, GT, GEQ, AND, OR, NEG, COMMA, RAPEN
expr_prime = ϵ | ASSIGN expr | LPAREN actuals_opt RPAREN	Assign, lparen, epsilon	SEMI, RPAREN, PLUS, MINUS, TIMES, DIVIDE, EQ, NEQ, LT, LEQ, GT, GEQ, AND, OR, NEG, COMMA, RPAREN
Expr_prime_prime = PLUS expr| MINUS expr| TIMES expr| DIVIDE expr| EQ expr| NEQ expr| LT expr| LEQ expr| GT expr| GEQ expr| AND expr| OR expr	Plus, minus, times, divide, EQ, NEQ, LT, LEQ, GT, GEQ, AND, OR, epsilon, Assign, Lparen	SEMI, RPAREN, PLUS, MINUS, TIMES, DIVIDE, EQ, NEQ, LT, LEQ, GT, GEQ, AND, OR, NEG, COMMA, RPAREN
Actuals_opt = actuals_list | epsilon	INTLiteral, true, false, id, minus, not, lparen, epsilon	rparen

actuals_list = expr actuals_list_prime	intLiteral, true, false, id, minus, not, lparen	rparen
Actuals_list_prime = COMMA expr actuals_list_prime | epsilon	COMMA, epsilon	rapren