| 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 | $\varepsilon$ , “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 = LITERAL expr_prime<br>\| TRUE expr_prime<br>\| FALSE expr_prime<br>\| ID expr_prime<br>\| MINUS expr %prec NEG expr_prime<br>\| NOT expr expr_prime<br>\| ID ASSIGN expr expr_prime<br>\| ID LPAREN actuals_opt RPAREN expr_prime<br>\| LPAREN expr RPAREN expr_prime | INTLiteral, true, false, id, minus, not, lparen | SEMI, RPAREN, NEG, PLUS, MINUS, TIMES, DIVIDE, EQ, NEQ, LT, LEQ, GT, GEQ, AND, OR, COMMA |
| expr_prime = PLUS expr expr_prime<br>\| MINUS expr expr_prime<br>\| TIMES expr expr_prime<br>\| DIVIDE expr expr_prime<br>\| EQ expr expr_prime<br>\| NEQ expr expr_prime<br>\| LT expr expr_prime<br>\| LEQ expr expr_prime | Plus, minus, times, divide, eq, neq, lt, leq, gt, geq, and, or | SEMI, RPAREN, NEG, PLUS, MINUS, TIMES, DIVIDE, EQ, NEQ, LT, LEQ, GT, GEQ, AND, OR, COMMA |

| | | |
|---|---|---|
| \| GT expr expr_prime<br>\| GEQ expr expr_prime<br>\| AND expr expr_prime<br>\| OR expr expr_prime<br>\| ϵ | | |
| 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 |