**Context Free Grammar**

**Legend:**

Blue → tokens

Red → different terminals

Purple → question here

1. Program → Function\_Statements Main\_Function

|Main\_Function

1. Main\_Function→ int main() Function\_Body

* Function\_Statements→ Function\_Statement Function\_Statements ‘
* Function\_Statements ‘→Function\_Statement Function\_Statements ‘ | ε

1. Function\_Statement→ Function\_Declaration Function\_Body
2. Function\_Body→ { Statements Return\_Statement }
3. Function\_Declaration→ Data\_Type Function\_Name (Parameters)

* Parameters → Parameter Parameters ’
* Parameters ’ → , Parameter Parameters ’ | ε

1. Parameter → Data\_Type Identifier
2. Function\_Name→ Identifier

* Statements → Statement Statements ’
* Statements ’ →Statement Statements ’ | ε

1. Statement → Assign\_Statement | Declaration\_Statement | Write\_Statement | Read\_Statement | Condition\_Statement | If Condition then Statements Else\_Clause|

Repeat\_Statement

1. Data\_Type → int | float | string
2. Assignment\_Statement → Identifier := Expression
3. Declaration\_Statement → Data\_Type ID\_Assign ;
4. ID\_Assign → Assignment\_Statements Identifiers

* Assignment\_Statements → Assignment\_Statement Assignment\_Statements ‘
* Assignment\_Statements ‘ → , Assignment\_Statement Assignment\_Statements ‘ | ε
* Identifiers → Identifier Identifiers ‘
* Identifiers ‘ → , Identifier Identifiers ‘ | ε

1. Write\_Statement → write Write\_Statement’

- Write\_Statement’ → Expression ; | endl ;

1. Read\_Statement → read Identifier ;
2. Return\_Statement → return Expression ;
3. Condition\_Statement → Condition Bool\_Op
4. Condition → Identifier Condition\_Operator Term

* Bool\_Op → Boolean\_Operator Condition Bool\_Op ‘
* Bool\_Op ‘ → Boolean\_Operator Condition Bool\_Op ‘ | ε

1. Term → Number | Identifier | Function\_Call
2. Function\_Call → Identifier (Identifiers)
3. Expression → String | Term | Equation

- Else\_Clause → else Statements end Else\_Clause’ | end Else\_Clause’

- Else\_Clause’ → elseif Condition then Statements end Else\_Clause’ | ε

1. Repeat\_Statement → repeat Statements until Condition\_Statements

* Equation → TermEq Equation‘
* Equation‘ → Add\_OP TermEq Equation ‘ | ε

1. Add\_OP → + | -

* TermEq → Factor TermEq‘
* TermEq‘ → Mult\_OP Factor TermEq‘ | ε
* Factor → Term Factor’

- Factor’ → (Equation) Factor’ | ε

1. Mult\_OP → \* | /