**<Program>** --> {<define>} {<VariableDeclaration>} {<FunctionDefinition>} <main>

**<define>** --> "definis" <ID> ":" (<NumLiteral> | <StrLiteral> | <CharLiteral>)

**<Main>** --> "demarre" <block>

**<ID>** --> ["\*"]<letter>{<digit>|<letter>}["["<ID> | <NumLiteral>"]"]

**<CharLiteral>** --> "'" (<letter>|<symbol>|<digit>)"'"

**<letter>** --> "A" | "B" | "C" | "D" |...| "Z" | "a" | "b" | "c" | "d" |...| "z" **<digit>** --> "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

**<symbol>** --> "|" | " " | "!" | "#" | "$" | "%" | "&" | "(" | ")" | "\*" | "+" | "," | "-" | "." | "/" | ":" | ";" | ">" | "=" | "<" | "?" | "@" | "[" | "\" | "]" | "^" | "\_" | "{" | "}" | "~"

**<NumLiteral>** --> ["+"|"-"]{<digit>}<digit>

**<StrLiteral>** --> """{<letter>|<digit>|<space>|<symbol>}"""

**<VariableDeclaration>** --> (<entierDeclaration> | <charDeclaration> | <arrayDeclaration>)"."

**<entierDeclaration>** --> "entier" <ID> [":" (<NumLiteral> | <ID>)] {"," <ID> [":" (<NumLiteral> | <ID>)]} "."

**<charDeclaration>** --> "car" <ID> [":" (<CharLiteral> | <ID>)] {"," <ID> [":" (<CharLiteral> | <ID>)]} "."

**<array declaration>** --> <entier array> | <char array>

**<entier array>** --> "entier" <ID>"["(<NumLiteral>|<ID>)"]" [":" "{"(<ID> | <NumLiteral>) {","<ID> | <NumLiteral>}"}"] "."

**<char array>** --> "car" <ID>"["(<NumLiteral>|<ID>)"]" [ ":" "{"(<ID> | <CharLiteral>) {","<ID> | <CharLiteral>} "}" ]"."

**<FunctionDefinition>** --> "fonction" <ID> "("[<parameter> {","<parameter>}]")" "renvoi" <DataType> <block>

**<parameter>** --> <DataType> <ID>

**<DataType>** --> "entier" | "car"

**<block>** --> "debut" {<VariableDeclaration>} {<statement>} "fin"

**<statement>** --> <SelectionStatement> | <RepitionStatement> | <AssignmentStatement> | <FunctionCall> | <ReturnStatement>

**<SelectionStatement>** --> "si" ("("<RelationalExpression>")" | "["<LogicalExpression>"]") <block> {"sinon si" ("("<RelationalExpression>")" | "["<LogicalExpression>"]") <block>} ["sinon" <block>]

**<RepetitionStatement>** --> "tantque" ("("<RelationalExpression>")" | "["<LogicalExpression>"]") <block>

**<AssignmentStatement>** --> <identifer> ":" (<arithmetic exp> | <NumLiteral> | <ID> | <CharLiteral> | <FunctionCall>) "."

**<ArithmeticExpression>** --> “[”<term> {(“+” | “-”) <term>}“]”"."

**<term>** --> <factor> {(“\*” | “/”) <factor>}

**<factor>** --> (<ID>|<NumLiteral>)

**<expression>** --> <RelationalExpression> | <LogicalExpression>

**<RelationalExpression>** --> (<ID> | <NumLiteral> | <CharLiteral>)("=" | ">" | "<" | "?") (<ID> | <NumLiteral> | <CharLiteral>)

**<LogicalExpression>** --> <andFirst> {“ou” < andFirst >}

**<andFirst>** --> [“pas”] <RelationalExpression> {“et” [“pas”]<RelationalExpression> }

**<function call>** --> "(" <ID>"(" [<ID>] ")" ")" "." | <PredefinedFunction>

**<PredefinF function>** --> "scanne""("<ID>")""." | "affiche""("<value>")""."

**<value>** --> <NumLiteral> | <StrLiteral> | <ID> | <CharLiteral>

**<space>** --> " "{" "}

**<ReturnStatement>** --> "renvoi" (<ID> | <NumLiteral> | <CharLiteral>) "."

Every non-terminal is enclosed in double quotes, for readability purposes we colored some non-terminals in red especially the punctuations that might interfere with the punctuations used in EBNF (e.g., parentheses used for alternation…).