

Complete Grammar

```
Program ::= Type IDENT ( ParamList ) Block
Block ::= <: ( Declaration ; | Statement ; ) * :>
ParamList ::= ε | NameDef ( , NameDef ) *
NameDef ::= Type IDENT | Type Dimension IDENT
Type ::= image | pixel | int | string | void | boolean
Declaration ::= NameDef | NameDef = Expr
Expr ::= ConditionalExpr | LogicalOrExpr
ConditionalExpr ::= ? Expr -> Expr , Expr
LogicalOrExpr ::= LogicalAndExpr ( ( | | || ) LogicalAndExpr ) *
LogicalAndExpr ::= ComparisonExpr ( ( & | && ) ComparisonExpr ) *
ComparisonExpr ::= PowExpr ( ( < | > | == | <= | >= ) PowExpr ) *
PowExpr ::= AdditiveExpr ** PowExpr | AdditiveExpr
AdditiveExpr ::= MultiplicativeExpr ( ( + | - ) MultiplicativeExpr ) *
MultiplicativeExpr ::= UnaryExpr ( ( * | / | % ) UnaryExpr ) *
UnaryExpr ::= ( ! | - | width | height ) UnaryExpr | PostfixExpr
PostfixExpr ::= PrimaryExpr ( PixelSelector | ε ) ( ChannelSelector | ε )
PrimaryExpr ::= STRING_LIT | NUM_LIT | IDENT | ( Expr ) | CONST | BOOLEAN_LIT |
    ExpandedPixelExpr
ChannelSelector ::= : red | : green | : blue
PixelSelector ::= [ Expr , Expr ]
ExpandedPixelExpr ::= [ Expr , Expr , Expr ]
PixelSelectorInLValue ::= [ Expr , Expr ]
Dimension ::= [ Expr , Expr ]
LValue ::= IDENT ( PixelSelector | ε ) ( ChannelSelector | ε )
Statement ::=
    LValue = Expr |
    write Expr |
    do GuardedBlock [] GuardedBlock* od |
    if GuardedBlock [] GuardedBlock* if |
    ^ Expr |
    BlockStatement |
GuardedBlock ::= Expr -> Block
BlockStatement ::= Block
```

Note: the rules with orange background were parsed in assignment 1.