~~program ::= IDENT block~~

~~program ::= IDENT param\_dec (~~ **~~,~~** ~~param\_dec )\* block~~

**Bold: defined in starter code**

1. **Parse**
2. **program** ::= IDENT program\_tail
3. program\_tail ::= block | param\_dec ( **,** param\_dec )\* block
4. **param\_dec** ::= ( KW\_URL | KW\_FILE | KW\_INTEGER | KW\_BOOLEAN) IDENT
5. **block** ::= **{** ( dec | statement) \* **}**
6. **dec** ::= ( KW\_INTEGER | KW\_BOOLEAN | KW\_IMAGE | KW\_FRAME) IDENT
7. **statement** ::= OP\_SLEEP expression **;** | whileStatement | ifStatement | chain **;** | assign **;**
8. assign ::= IDENT ASSIGN expression
9. **chain** ::= chainElem arrowOp chainElem ( arrowOp chainElem)\*
10. whileStatement ::= KW\_WHILE **(** expression **)** block
11. ifStatement ::= KW\_IF **(** expression **)** block
12. arrowOp ∷= ARROW | BARARROW
13. **chainElem** ::= IDENT | filterOp arg | frameOp arg | imageOp arg
14. filterOp ::= OP\_BLUR | OP\_GRAY | OP\_CONVOLVE
15. frameOp ::= KW\_SHOW | KW\_HIDE | KW\_MOVE | KW\_XLOC | KW\_YLOC
16. imageOp ::= OP\_WIDTH | OP\_HEIGHT | KW\_SCALE
17. **arg** ::= ε | **(** expression ( , expression)\* **)**
18. **expression** ∷= term ( relOp term)\*
19. **term** ∷= elem ( weakOp elem)\*
20. **elem** ∷= factor ( strongOp factor)\*
21. **factor** ∷= IDENT | INT\_LIT | KW\_TRUE | KW\_FALSE | KW\_SCREENWIDTH | KW\_SCREENHEIGHT | **(** expression **)**
22. relOp ∷= LT | LE | GT | GE | EQUAL | NOTEQUAL
23. weakOp ∷= PLUS | MINUS | OR
24. strongOp ∷= TIMES | DIV | AND | MOD

Defined by Sanders:

void parse() throws SyntaxException {

void expression() throws SyntaxException {

void term() throws SyntaxException {

void elem() throws SyntaxException {

void factor() throws SyntaxException

void block() throws SyntaxException {

void program() throws SyntaxException {

void paramDec() throws SyntaxException {

void dec() throws SyntaxException {

void statement() throws SyntaxException {

void chain() throws SyntaxException {

void chainElem() throws SyntaxException {

void arg() throws SyntaxException {

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1** | 1 | **9** | 1 | **17** | 1 |
| **2** | 2 | **10** | 1 | **18** | 1 |
| **3** | 4 | **11** | 2 | **19** | 1 |
| **4** | 1 | **12** | 4 | **20** | 7 |
| **5** | 4 | **13** | 3 | **21** | 6 |
| **6** | 5 | **14** | 5 | **22** | 3 |
| **7** | 1 | **15** | 3 | **23** | 4 |
| **8** | 1 | **16** | 2 |  |  |

Implementation and testing done: 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13,12, 11, 10, 9, 8, 7, 6, 5, 4, 3,2,1

1. program ::= IDENT program\_tail
   1. Predict (program ::= IDENT program\_tail) = {IDENT}
2. program\_tail ::= block | param\_dec ( **,** param\_dec )\* block
   1. Predict(program\_tail ::= block) = { **{** }
   2. Predict (program\_tail ::= param\_dec ( **,** param\_dec )\* block) = { KW\_URL, KW\_FILE, KW\_INTEGER, KW\_BOOLEAN }
3. param\_dec ::= ( KW\_URL | KW\_FILE | KW\_INTEGER | KW\_BOOLEAN) IDENT
   1. Predict(param\_dec ::= KW\_URL IDENT) = {KW\_ URL}
   2. Predict(param\_dec ::= KW\_FILE IDENT) = {KW\_ FILE}
   3. Predict(param\_dec ::= KW\_INTEGER IDENT) = {KW\_ INTEGER}
   4. Predict(param\_dec ::= KW\_BOOLEAN IDENT) = {KW\_BOOLEAN}
4. block ::= **{** ( dec | statement) \* **}**
   1. Predict(block ::= **{** ( dec | statement) \* **}) =** {**{** }
5. dec ::= ( KW\_INTEGER | KW\_BOOLEAN | KW\_IMAGE | KW\_FRAME) IDENT

FIRST(dec) = { union of all predict sets} (to be used in rule 4)

* 1. Predict(dec ::= KW\_INTEGER IDENT) = {KW\_INTEGER}
  2. Predict(dec ::= KW\_BOOLEAN IDENT) = {KW\_BOOLEAN}
  3. Predict(dec ::= KW\_IMAGE IDENT) = {KW\_IMAGE}
  4. Predict(dec ::= KW\_FRAME IDENT) = {KW\_FRAME}

1. statement ::= OP\_SLEEP expression **;** | whileStatement | ifStatement | chain **;** | assign **; (NOT LL1; it is LL2)**

FIRST(statement) = { union of all predict sets} ( to be used for rule 4)

* 1. Predict(statement ::= OP\_SLEEP expression **;**) = {OP\_SLEEP}
  2. Predict(statement ::= whileStatement) = {KW\_WHILE}
  3. Predict(statement ::= ifStatement) = {KW\_IF}
  4. Predict(statement ::= chain **;**) = {IDENT, OP\_BLUR,OP\_GRAY ,OP\_CONVOLVE , KW\_SHOW , KW\_HIDE , KW\_MOVE , KW\_XLOC , KW\_YLOC, OP\_WIDTH , OP\_HEIGHT , KW\_SCALE }
  5. Predict(statement ::= assign **;**) = {IDENT}

1. assign ::= IDENT ASSIGN expression
   1. PREDICT(assign ::= IDENT ASSIGN expression ) = {IDENT}
2. chain ::= chainElem arrowOp chainElem ( arrowOp chainElem)\*
   1. Predict(chain ::= chainElem arrowOp chainElem ( arrowOp chainElem)\*) = { IDENT, OP\_BLUR,OP\_GRAY ,OP\_CONVOLVE , KW\_SHOW , KW\_HIDE , KW\_MOVE , KW\_XLOC , KW\_YLOC, OP\_WIDTH , OP\_HEIGHT , KW\_SCALE }
3. whileStatement ::= KW\_WHILE **(** expression **)** block
   1. Predict(whileStatement ::= KW\_WHILE **(** expression **)** block) = {KW\_WHILE}
4. ifStatement ::= KW\_IF **(** expression **)** block
   1. PREDICT(ifStatement ::= KW\_IF **(** expression **)** block) = {KW\_IF}
5. arrowOp ∷= ARROW | BARARROW

First(Arrowop) = Union of predict sets ( rule 8)

* 1. PREDICT(arrowOp ∷= ARROW) = {ARROW}
  2. PREDICT(arrowOp ∷= BARARROW) = {BARARROW}

1. chainElem ::= IDENT | filterOp arg | frameOp arg | imageOp arg
   1. PREDICT(chainElem ::= IDENT) = {IDENT}
   2. PREDICT(chainElem ::= filterOp arg) = { OP\_BLUR ,OP\_GRAY , OP\_CONVOLVE }
   3. PREDICT(chainElem ::= frameOp arg) = { KW\_SHOW , KW\_HIDE , KW\_MOVE , KW\_XLOC , KW\_YLOC }
   4. PREDICT(chainElem ::= imageOp arg) = { OP\_WIDTH , OP\_HEIGHT , KW\_SCALE }
2. filterOp ::= OP\_BLUR | OP\_GRAY | OP\_CONVOLVE
   1. Predict(filterOp ::= OP\_BLUR) = { OP\_BLUR }
   2. Predict(filterOp ::= OP\_GRAY) = { OP\_GRAY }
   3. Predict(filterOp ::= OP\_CONVOLVE) = { OP\_CONVOLVE }
3. frameOp ::= KW\_SHOW | KW\_HIDE | KW\_MOVE | KW\_XLOC | KW\_YLOC
   1. PREDICT(frameOp ::= KW\_SHOW) = {KW\_SHOW}
   2. PREDICT(frameOp ::= KW\_HIDE) = {KW\_HIDE}
   3. PREDICT(frameOp ::= KW\_MOVE) = {KW\_MOVE}
   4. PREDICT(frameOp ::= KW\_XLOC) = {KW\_XLOC}
   5. PREDICT(frameOp ::= KW\_YLOC) = {KW\_YLOC}
4. imageOp ::= OP\_WIDTH | OP\_HEIGHT | KW\_SCALE
   1. PREDICT(imageOp ::= OP\_WIDTH) = { OP\_WIDTH }
   2. PREDICT(imageOp ::= OP\_HEIGHT) = { OP\_HEIGHT }
   3. PREDICT(imageOp ::= KW\_SCALE) = { KW\_SCALE }
5. arg ::= ε | **(** expression ( **,** expression)\* **)**
   1. Predict(arg::= epsilon) = {ARROW, BARARROW, **;** }
   2. Predict(arg::= **(** expression ( **,** expression)\* **)**  ) = { **(** }
6. expression ∷= term ( relOp term)\* sigma\* **while(token ∈ FIRST(σ)) { parse σ; }**
   1. PREDICT(expression ∷= term ( relOp term)\*) = { IDENT , INT\_LIT , KW\_TRUE , KW\_FALSE , KW\_SCREENWIDTH , KW\_SCREENHEIGHT , **(** }
7. term ∷= elem ( weakOp elem)\*
   1. PREDICT(term ∷= elem ( weakOp elem)\* ) = { IDENT , INT\_LIT , KW\_TRUE , KW\_FALSE , KW\_SCREENWIDTH , KW\_SCREENHEIGHT , **(** }
8. elem ∷= factor ( strongOp factor)\*
   1. PREDICT(elem ∷= factor ( strongOp factor)\*) = { IDENT , INT\_LIT , KW\_TRUE , KW\_FALSE , KW\_SCREENWIDTH , KW\_SCREENHEIGHT , **(** }
9. factor ∷= IDENT | INT\_LIT | KW\_TRUE | KW\_FALSE | KW\_SCREENWIDTH | KW\_SCREENHEIGHT | **(** expression **)**
   1. PREDICT(factor ∷= IDENT) = { IDENT }
   2. PREDICT(factor ∷= INT\_LIT) = { INT\_LIT }
   3. PREDICT(factor ∷= KW\_TRUE) = { KW\_TRUE }
   4. PREDICT(factor ∷= KW\_FALSE) = { KW\_FALSE }
   5. PREDICT(factor ∷= KW\_SCREENWIDTH) = { KW\_SCREENWIDTH }
   6. PREDICT(factor ∷= KW\_SCREENHEIGHT) = { KW\_SCREENHEIGHT }
   7. PREDICT(factor ∷= **(** expression **)**) = { **(** }
10. relOp ∷= LT | LE | GT | GE | EQUAL | NOTEQUAL

First(relOp) = Union of predict sets

* 1. PREDICT (relOp ∷= LT) = {LT}
  2. PREDICT (relOp ∷= LE) = {LE}
  3. PREDICT (relOp ∷= GT) = {GT}
  4. PREDICT (relOp ∷= GE) = {GE}
  5. PREDICT (relOp ∷= EQUAL) = {EQUAL}
  6. PREDICT (relOp ∷= NOTEQUAL) = {NOTEQUAL}

1. weakOp ∷= PLUS | MINUS | OR

First(weakOp) = Union of Predict Sets

* 1. PREDICT(weakOp ∷= PLUS) = {PLUS}
  2. PREDICT(weakOp ∷= MINUS) = {MINUS}
  3. PREDICT(weakOp ∷= OR) = {OR}

1. strongOp ∷= TIMES | DIV | AND | MOD

First(strongOp) = Union of Predict Sets

* 1. PREDICT(strongOp ∷= TIMES) = {TIMES}
  2. PREDICT(strongOp ∷= DIV) = {DIV}
  3. PREDICT(strongOp ∷= AND) = {AND}
  4. PREDICT(strongOp ∷= MOD) = {MOD}

**5 rules (+ tests) a day**

**Conversions:**

1. **For each non-terminal A, we will have a method void A().**
2. **σ1 | σ 2 | ... σn (where none are ε)**

**becomes code fragment**

**if (token ∈ PREDICT(l.h.s.::= σ1 )) parse σ1 ;**

**else if (token ∈ PREDICT(l.h.s::= σ 2 ))**

**parse σ 2;**

**...**

**else if (token ∈ PREDICT(l.h.s. ::= σn ))**

**parse σn**

**else error()**

1. **σ1 | σ 2 | ... σn | ε**

**becomes**

**if (token ∈ PREDICT(l.h.s.::= σ1 )) parse σ1 ;**

**else if (token ∈ PREDICT(l.h.s::= σ2 ))**

**parse σ2;**

**...**

**else if (token ∈ PREDICT(l.h.s. ::= σn ))**

**parse σn ;**

**else just return //this branch matches ε**

1. **σ1 σ 2 ... σn**

**becomes**

**parse σ1;**

**parse σ2;**

**…**

**parse σn**

1. **σ\***

**becomes**

**while(token ∈ FIRST(σ)) { parse σ; }**

1. **c (where a is a terminal symbol)**

**becomes**

**match(c);**

**where**

**match(c)  
{ if ( current token is c )**

**{ get next token from scanner;}**

**else error;**

**}**