PLATYPUS LANGUAGE SPECIFICATION

## -2.10 OPERATOR:

FIRST(operator)={==,<,>}

<operator>-> ==|<|>

## 3 STATEMENTS:

-OPTIONAL STATEMENTS:

FIRST set:{AVID\_T,SVID\_T,KW\_T (no PLATYPUS,ELSE,THEN,REPEAT),AVID\_T,SVID\_T,and 

<opt\_statements>->statements()

-3.2 STATEMENTS:

FIRST(statements)={AVID,SVID,IF,USING,INPUT,OUTPUT}

<statements>->

<statement><statements'>

- 3.2 STATEMENTS’:

FIRST(STATEMENTS')={AVID,SVID,IF,USING,INPUT,OUTPUT,}

<statements'>->

<statement><statements'>|

-3.2 STATEMENT:

FIRST(statemetnt)={AVID,SVID,IF,USING,INPUT,OUTPUT}

<statements>->

<assignment statement>

|<selection statement>

|<iteration statement>

|<input statement>

|<output statement>

-3.2.1 ASSIGNMENT\_STATEMENT:

FIRST(assignment statement)={AVID,SVID}

<assignment statement>->

<assignment expression>;

-3.2.1 ASSIGNMENT EXPRESSION:

FIRST(assignment expression)={AVID,SVID}

<assignment expression>->

AVID=<artimetic expression>

|SVID=<string expression>

-3.2.2 SELECTION STATEMENT:

FIRST(assignment expression)={IF}

<selection statement>->

IF(<conditional expression>)THEN<opt\_statements>

ELSE{<opt\_statements>};

-3.2.3 ITERATION STATEMENT:

FIRST(iteration statement)={USING}

<iteration statement>->

USING(<assignment expression>,<conditional expression>,<assignment expression>)

REPEAT{

<opt\_statement> };

-3.2.4 INPUT STATEMENT:

FIRST(input statement)={INPUT}

<input statement> ->

INPUT(<variable list>);

-3.2.5 OUTPUT\_STATEMETNT:

FIRST(output statement)={OUTPUT}

<output statement>->

OUTPUT(<output list>);

-VARIABLE LIST:

FIRST(variable list)={variable identifier}

<variable list>->

<variable identifier><variable list'>

-VARIABLE LIST’:

FIRST(variable list')={, ,}

<variable list'> ->

,<variable identifier> <variable list'>|

-OUTPUT LIST:

FIRST(output list)={variable identifier,STR\_T,}

<output list> ->

<variable list>|STR\_T|e

-3.3.1 ARITHMETIC EXPRESSION:

FIRST(arithmetic expression)={-,+,AVID\_T,FPL\_T,INL\_T,(}

<arithemetic expression> ->

<unary arithemetic expression>

|<additive arithemetic expression>

-3.3.1 UNARY ARITHMETIC EXPRESSION:

FIRST(unary arithemetic expression)={-,+}

<unary arothemetic expression>->

- <primary arithemetic expression>

| + <primary arithemetic expression>

-3.3.1 ADDITIVE ARITHMETIC EXPRESSION:

FIRST(additive arithemtic expression)={AVID\_T,FPL\_T,INL\_T,(}

<additive arithmetic expression>->

<multiplicative arithmetic expression> <additive arithmetic expression'>

## - 3.3.1 ADDITIVE ARITHMETIC EXPRESSION’:

FIRST(additive arithmetic expression')={+,-,}

<additive arithmetic expression'>->

+ <multiplicative arithmetic expression> <additive arithmetic expression'>

|- <multiplicative arithmetic expression> <additive arithmetic expression'>

| 

## -3.3.1 MULTIPLICATIVE ARITHMETIC EXPRESSION:

FIRST(multiplicative arithmetic expression)={AVID\_T,FPL\_T,INL\_T,(}

<multiplicative arithmetic expression>->

<primary arithmetic expression> <multuplicative arithmetic expression'>

## - 3.3.1 MULTIPLICATIVE ARITHMETIC EXPRESSION’:

FIRST(multiplicative arithmetic expression')={\*,/,}

<multiplicative arithmetic expression'> ->

\* <primary arithmetic expression> <multuplicative arithmetic expression'>

| /<primary arithmetic expression> <multuplicative arithmetic expression'>

| e

## -3.3.1 PRIMARY ARITHMETIC EXPRESSION:

FIRST(primary arithmetic expression)={AVID\_T,FPL\_T,INL\_T,(}

<primary arithmetic expression> ->

AVID\_T

| FPL\_T

| INL\_T

| (<arithmetic expression>)

## -3.3.2 STRING EXPRESSION:

FIRST (string expression)={SVID\_T,STR\_T}

<string expression> ->

<primary string expression> <string expression'>

## -3.3.2 STRING EXPRESSION’:

FIRST(string expression')={#,}

<string expression'>->

<primary string expression> <string expression'>|e

## -3.3.2 PRIMARY STRING EXPRESSION:

FIRST(primary string expression={SVID\_T,STR\_T}

<primary string expression> ->

SVID\_T|STR\_T

## -3.3.3 CONDITIONAL EXPRESSION:

FIRST(conditional expression)={AVID\_T,FPL\_T,INL\_T,SVID\_T,STR\_T}

<conditional expression> ->

<logical OR expression>

## -3.3.3 LOGICAL OR:

FIRST(logical OR expression)={AVID\_T,FPL\_T,INL\_T,SVID\_T,STR\_T}

<logical OR expression>->

<logical AND expression>

|<logical OR expression>

## - 3.3.3 LOGICAL OR’:

FIRST(logical OR expression')={.OR.,}

<logical OR expression'> ->

.OR. <logical AND expression> <logical OR expression>|

## -3.3.3 LOGICAL AND:

FIRST(logical AND expression)={AVID\_T,FPL\_T,INL\_T,SVID\_T,STR\_T}

<logical AND expression> ->

<relational expression> <logical AND expression'>

## -3.3.3 LOGICAL AND’:

FIRST (logical AND expression')={.AND.,}

<logical AND expression'> ->

.AND. <relational expression><logical AND expression'>|

## -3.3.4 RELATIONAL EXPRESSION:

FIRST(relational expression)={AVID\_T,FPL\_T}

<relational expression>->

<primary a\_relational expression> <operator> <primary a\_relational expression>

|<primary s\_relational expression> <operator> <primary s\_relational expression>

## -3.3.4 PRIMARY A RELATIONAL EXPRESSION :

FIRST (primary a\_relational expression)={AVID\_T,FPL\_T,INL\_T}

<primary a\_relational expression>->

AVID\_T | FPL\_T | INL\_T

## -3.3.4 PRIMARY S RELATIONAL EXPRESSION :

FIRST (primary s\_relational expression)={SVID\_T,STR\_T}

<primary s\_relational expressino>->

<primary string expression>