FIRST(<program>) -> { PLATYPUS }

FIRST(<opt\_statements>) = { AVID\_T, SVID\_T, IF, WHILE, READ, WRITE, e }

FIRST (<statement>) = { AVID\_T, SVID\_T, IF, WHILE, READ, WRITE}

FIRST (<statements’>) = { AVID\_T, SVID\_T, IF, WHILE, READ, WRITE, e }

FIRST(<assignment statement>) = { AVID\_T, SVID\_T }

FIRST(<assignment expression>) = { AVID\_T, SVID\_T }

FIRST (<selection statement>) = { IF }

FIRST (<iteration statement>) = { WHILE }

FIRST(<pre-condition>) = { TRUE, FALSE }

FIRST (<input statement>) = { READ }

FIRST (<variable list>) = { AVID\_T, SVID\_T }

FIRST (<variable list’>) = { , , e }

FIRST (<variable identifier>) = { AVID\_T, SVID\_T }

FIRST(<output statement>) = { WRITE }

FIRST (<output\_list>) = {AVID\_T, SVID\_T, STR\_T, e }

FIRST (<input\_list>) = {AVID\_T, SVID\_T, STR\_T, e }

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

FIRST (<unary arithmetic expression>) = { -, +}

FIRST (<additive arithmetic expression>) = { AVID\_T, FPL\_T, INL\_T, ( }

FIRST (<additive arithmetic expression’>) = { +, -, e}

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

FIRST (<multiplicative arithmetic expression’>) = { \*, / , e}

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

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

FIRST ( <string expression'> ) = { <<, e}

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

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

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

FIRST (<logical OR expression’>) = { .OR. , e }

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

FIRST (<logical AND expression’>) = {.AND. , e }