```
pgm -> d-list => first= int void float => follow= $
d-list -> dclartn A  => first= int void float => follow= $
A -> dclartn A | @  => first= int void float @ => follow= $
dclartn -> type-specfr id B  => first= int void float => follow= int void float
B -> var-dclartn | fun-dclartn  => first= ; [ ( => follow= int void float
var-dclartn -> ; | [ num ] ;  => first= ; [ => follow= int void float
fun-dclartn -> ( params ) compnd-stmt  => first= (  => follow= int void float
type-specfr -> int | void | float  => first= int void float => follow= id
params -> param-list | void  => first= int void float => follow= )
param-list -> param D  => first= int void float => follow= )
D -> , param D | @  => first= , @ => follow= )
param -> type-specfr id E  => first= int void float  => follow= , )
E -> [ ] | @  => first= [ @  => follow= , )
compnd-stmt -> { local-dclartn statmnt-list }  => first= { => follow= int void
float
local-dclartn -> F  => first= int void float => follow= ( id num if while return
{ }
F -> type-specfr id var-dclartn F | @  => first= int void float  => follow=
statmnt-list -> G  => first= ( id num if while return {  => follow= }
G -> statmnt G | @ => first= @ ( id num if while return {  => follow= }
statmnt -> exprsn-stmt => first= ( id num if while return {  => follow= ( id num
if while return { }
statmnt -> selctn-stmt
statmnt -> itertn-stmt
statmnt -> retrn-stmt
statmnt -> compnd-stmt
exprsn-stmt -> exprsn ; | ;  => first= ( id num  => follow= ( id num if while
return { }
selctn-stmt -> if ( exprsn ) statmnt H  => first= if  => follow= ( id num if
while return { }
H -> else statmnt | @  => first= else @  => follow= ( id num if while return { }
itertn-stmt -> while ( exprsn ) statmnt  => first= while  => follow= ( id num if
while return { }
retrn-stmt -> return exprsn-stmt  => first= return  => follow= ( id num if while
return { }

these are new rules

exprsn -> ( exprsn ) three  => first= ( id num  => follow= , ; ] )
exprsn -> num three
exprsn -> id two
two -> one relop two2 | ( args ) three | three => first= @ [ = (  => follow= , ;
] )
three -> temp3 temp2 temp  => first= @ * / + - < > = !  => follow= ; ] )
temp3 -> @ | mulop factor temp3  => first= @ * /  => follow= + - < > = ! ; ] )
temp2 -> @ | addop term temp2  => first= @ + -  => follow= < > = ! ; ] )
temp -> @ | relop addexp temp => first= @ < > = !  => follow= ; ] )
two2 -> three | exprsn  => first= @ * / + - < > = ! ( id num  => follow= ; ] )
relop -> @ | < L | > L | = L | ! =  => first= @ < > = !  => follow= ( id num
L -> = | @ => first= = @  => follow= ( id num
addexp -> term temp2  => first= ( id num  => follow= < > = ! ; ] )
addop -> + | -  => first= + -  => follow= ( id num
term -> factor temp3  => first= ( id num  => follow= + - < > = ! ; ] )
mulop -> * | /  => first= * /  => follow= ( id num
factor -> ( exprsn O )  => first= ( id num  => follow= * / + - < > = ! ; ] )
factor -> id factor | id one
factor -> num
one -> @ | [ exprsn ]  => first= @ [  => follow= * / + - < > = ! ; ] )
args -> @ | arg-list => first= @ ( id num  => follow= )
arg-list -> exprsn O => first= ( id num  => follow= )
O -> @ | , exprsn O => first=@ ,  => follow= )

```

55
56