<PROGRAMA>

S -> ( program id S' S'' MAIN )

S' -> DECLARACIONVARIABLLES | epsilon

S'' -> DECLARACIONFUNCIONES | epsilon

<DECLARACIONVARIABLLES>

S -> ( vars S' )

S' -> DECLARE S' | epsilon

<DECLARACIONFUNCIONES>

S -> ( functions S' )

S' -> FUNCION S' | epsilon

<MAIN>

S -> ( main S' )

S' -> BLOQUE S' | epsilon

<IMPRIMIR>

S -> ( print S' )

S' -> cte\_ctar | EXPRESION | LISTFUNCITONS | LAMBDA

<DECLARE>

S -> ( declare id S' )

S' -> DEFINIRLISTA | DEFINIRCTE

<DEFINIRLISTA>

S -> '( )

S' -> DEFINIRCTE S' | epsilon

<DEFINIRCTE>

S -> cte\_int | cte\_float | cte\_char

<LISTA>

S -> id | '( S' ) //aqui se deberia de agregar el poder poner una lista sin declarar

S' -> cte\_int S' | cte\_float S' | cte\_char | epsilon

<VARCTE>

S -> id // id2

S -> cte\_int // 1

S -> cte\_float // 1.1

<LISTFUNCITONS>

S -> RETURNELEMENT | RETURNLIST

<RETURNLIST>

S -> ( S' LISTA ) | APPEND | LISTA | CREATELIST | MAP | LLAMADA | FILTER

S' -> cdr

<RETURNELEMENT>

S -> ( S' RETURNLIST ) | #t | #f

S' -> car | length | null? | list? | empty?

<APPEND>

S -> ( append RETURNLIST RETURNLIST S' )

S' -> RETURNLIST S' | epsilon

<CREATELIST> //funcion list

S -> ( list s' )

S' -> EXPRESION S' | epsilon

<MAP>

S -> ( map S'' RETURNLIST S' )

S' -> RETURNLIST S' | epsilon

S'' -> ( lambda (PARAM) BLOQUE )

<FILTER>

S -> ( filter S' RETURNLIST )

S' -> ( lambda (PARAM) BLOQUE ) | even? | int? | float?

------------------------------------------------------------

Son equivalentes

~~<EXP>~~

~~S -> ( + EXP EXP )~~

~~S -> ( - EXP EXP )~~

~~S -> ( \* EXP EXP )~~

~~S -> ( / EXP EXP )~~

~~S -> ( + VARCTE )~~

~~S -> ( - VARCTE )~~

~~S -> VARCTE~~

~~S -> LLAMADA~~

~~S -> RETURNELEMENT~~

<EXP>

S -> ( SIGNOS1 EXP EXP )

S -> ( SIGNOS2 EXP EXP )

S -> ( SIGNOS1 VARCTE )

S -> VARCTE

S -> LLAMADA

S -> RETURNELEMENT

<SIGNOS1>

S -> + | -

<SIGNOS2>

S -> \* | /

------------------------------------------------------------------------

<EXPRESION> // para comparativas

S -> EXP

S -> ( SIGNOSRELACIONALES EXP EXP )

S -> EXPRESIONESUNARIAS

<SIGNOSRELACIONALES>

S -> > | < | != | =

<EXPRESIONESUNARIAS>

S -> ( S' EXP) | #t | #f

S' -> even? | int? | float? | list? | null? | empty?

// ni los predicados ni el #t y #f estan bien los tokens

<CONDICION>

S -> ( if EXPRESION BLOQUE BLOQUE )

<BLOQUE>

S -> CONDICION | IMPRIMIR | EXPRESION | LLAMADA | LAMBDA | LISTFUNCITONS

<TIPO>

S -> int

S -> float

S -> char

<PARAM>

S -> id S | epsilon

<FUNCION>

S -> ( define ( id PARAM ) BLOQUE )

<LLAMADA> //llamada de funcion

S -> ( id S' )

S' -> EXPRESION S' | LISTFUNCITONS S' | epsilon

<LAMBDA>

S -> ( ( lambda (PARAM) BLOQUE ) S' )

S' -> EXPRESION S' | LISTFUNCITONS S' | epsilon