Terminales = { if let id input alert eof ( ; ) , { : } entero number boolean function string return cadena switch case default break = || |= == + - / }

NoTerminales = { E 1 R 2 U 3 G V 4 Z O S D L Q X B Y M N T F H A K C P }

Axioma = P

Producciones = {

// 1

E -> R 1 {

E.tipo := if(1.tipo=void)

then R.tipo

elseif((R.tipo == 1.tipo))

then R.tipo

Else

tipo\_Error

}

// 2

1 -> || R 1 {

1.tipo := {

if(1.tipo=void)

then R.tipo

elseif(R.tipo = boolean and 1.tipo = boolean)

then R.tipo

Else

tipo\_Error

}

// 3

1 -> lambda {1.tipo:= void}

// 4

R -> U 2{

{

2.tipo := {

if(2.tipo=void)

then U.tipo

elseif(U.tipo and 2.tipo)

then U.tipo

Else

tipo\_Error

}

// 5

2 -> == U 2

2.tipo := {

if(2.tipo=void)

then U.tipo

elseif(U.tipo= boolean and 2.tipo = boolean)

then U.tipo

Else

tipo\_Error

}

// 6

2 -> lambda {2.tipo:=void}

// 7

U -> V 3 {

U.tipo := {

if(3.tipo=void)

then R.tipo

elseif(V.tipo = 1.tipo)

Then V.tipo

Else

tipo\_Error

}

}

// 8

3 -> G V 3 {

3.tipo := {

if(3.tipo=void and V.tipo = entero)

Then V.tipo

elseif(V.tipo = entero and 3.tipo = entero)

then V.tipo

Else

tipo\_Error

}

// 9

3 -> lambda { 3.tipo=void}

// 10

G -> +{}

// 11

G -> -{}

// 12

V -> Z 4

{

V.tipo := {

if(4.tipo=void and Z.tipo = entero)

Then Z.tipo

elseif(Z.tipo = entero and 4.tipo = entero)

then Z.tipo

Else

tipo\_Error

}

// 13

4 -> / Z 4

{

V.tipo := {

if(4.tipo=void and Z.tipo = entero)

Then Z.tipo

elseif(Z.tipo = entero and 4.tipo = entero)

then Z.tipo

Else

tipo\_Error

}

}

// 14

4 -> lambda {4.tipo := void}

// 15

Z -> id O {Z.tipo:= buscarTSTipo(id.pos)}

// 16

Z -> ( E ) {Z.tipo := E.tipo}

// 17

Z -> entero {Z.tipo := ent}

// 18

Z -> cadena {Z.tipo := string}

// 19

O -> lambda {O.tipo := void}

// 20

O -> ( L ) { O.tipo := L.tipo}

// 21

S -> id D ; {id.tipo:= buscarTSTipo(id.pos), S.tipo := if(id.tipo = D.tipo) then tipo\_ok else tipo\_error}

// 22

S -> alert ( E ) ; {S.tipo := if(E.tipo = boolean) then boolean elsif (E.tipo = ent) then entero else error}

// 23

S -> input ( id ) ; {id.tipo := buscarTSTipo(id.pos), S.tipo := if(id.tipo € {cadena, entero} ) then tipo\_ok, else tipo\_error; }

// 24

S -> return X ; {S.tipo :=

// 25

D -> = E {D.tipo := E.tipo}

// 26

D -> |= E {D.tipo := if(E.tipo = boolean) then boolean else error}

// 27

D -> ( L ) {D.tipo := L.tipo}

// 28

L -> E {if(E.tipo = buscarTS(listTipoParametro(i)) then tipo\_ok else tipo\_error} Q

// 29

L -> lambda { L.tipo := void}

// 30

Q -> , E {if(E.tipo == buscarTS(listTipoParametro(i)) then tipo\_ok else tipo\_error} Q

// 31

Q-> lambda { Q.tipo := void}

// 32

X -> E {X.tipo := E.tipo}

// 33

X -> lambda {X.tipo := void}

// 34

B -> if ( E ) S {B.tipo := if (E.tipo = boolean) then tipo\_ok else tipo\_error}

// 35

B -> let T id ; {id.tipo := T.tipo,, insertarTSTipoYdesp(id.pos,id.tipo, desp),id.desp = desp + T.ancho}

// 36

B -> S

// 37

B -> switch ( E ) { Y } {B.tipo := if (E.tipo = entero) then tipo\_ok else tipo\_error}

// 38

Y -> case entero : C M

// 39

M -> break ; N

// 40

M -> N

// 41

N -> Y

// 42

N -> default : C

// 43

N -> lambda {C.tipo := void}

//44

T -> number {T.tipo := ent, T.ancho:=2}

//45

T-> boolean {T.tipo := boolean, T.ancho:=1}

//46

T -> string {T.tipo := ent, T.ancho:=string.length}

//47

F -> function H {ZonaDec=true} id {TS:= crearTS();InsertEt(id.pos, etiqueta); insertValRet(id.pos, H.tipo) }

( A ) { InsertNParam(A.listParamTipo.size());InsertTipoParam(A.listParamTipo); TSact := TS;

Push(TS);desp:=0} { C }

//48

H -> T {H,tipo := T.tipo}

//49

H -> lambda {H.tipo := void}

// 50

A -> T id {K.listParamTipo.add(id.tipo)} K { id.tipo:= T.tipo; id.desp := T.ancho + desp; insertTsNueva(id.pos,id.tipo, id.desp); A.listParamTipo := K.listParamTipo}

// 51

A -> lambda {A.tipo=void}

// 52

K -> , T id { K.listParamTipo.add(id.tipo) ; insertTsNueva(id.pos,id.tipo, id.desp)} K { id.tipo:= T.tipo; id.desp := T.ancho + desp; ; }

//53

K -> lambda {K.tipo:= void}

// 54

C -> B C

//55

C -> lambda {C.tipo := void}

//56

P -> B P

//57

P -> F P

// 58

P -> lambda

// 59

P -> eof

}