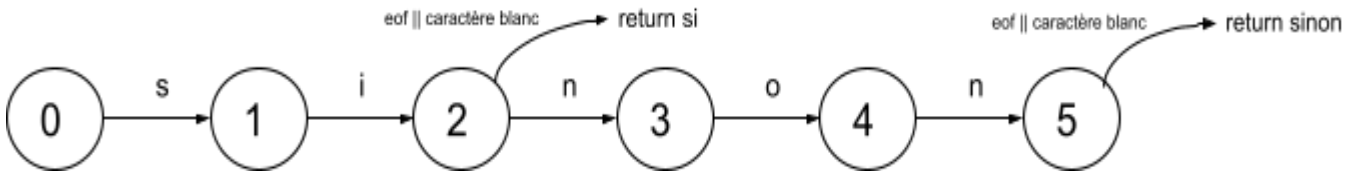
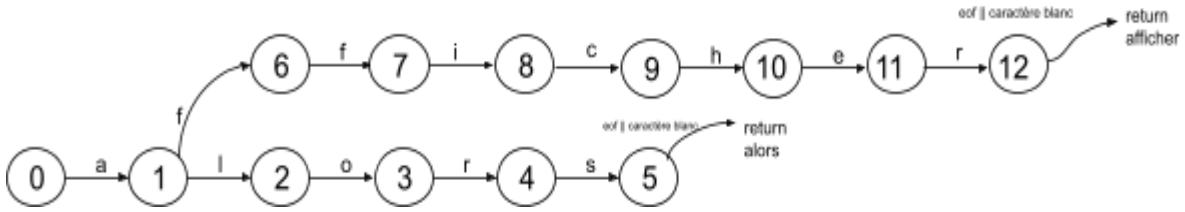
Analyse Lexicale

Les automates :

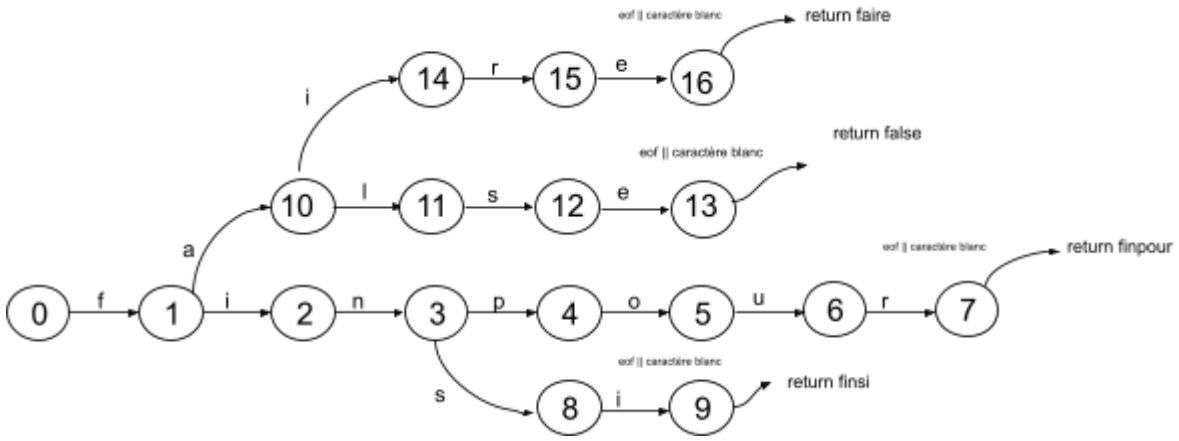
si sinon :



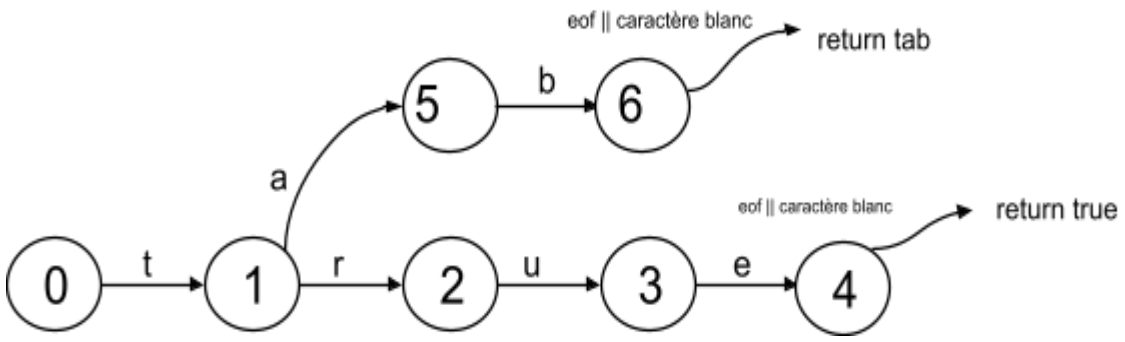
alors / afficher :



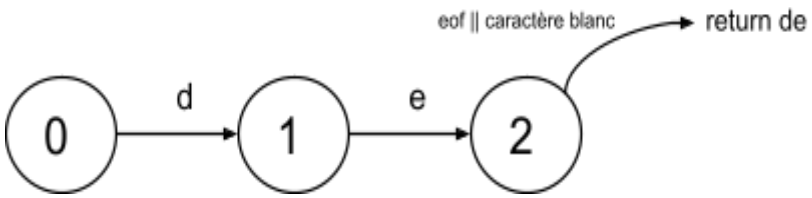
finpour / finsi / false / faire :



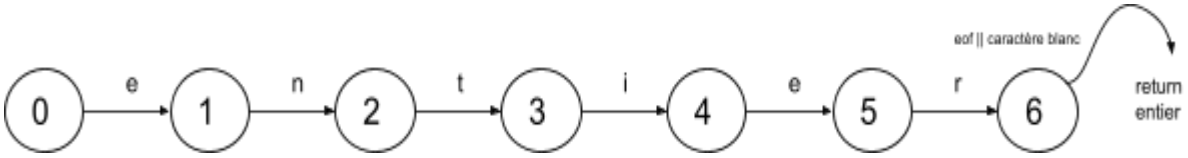
true / tab :



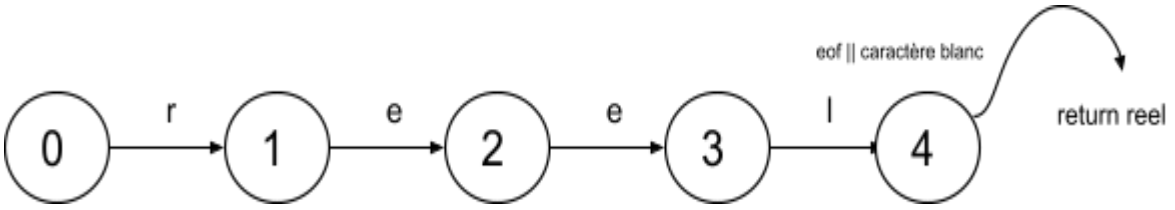
de :



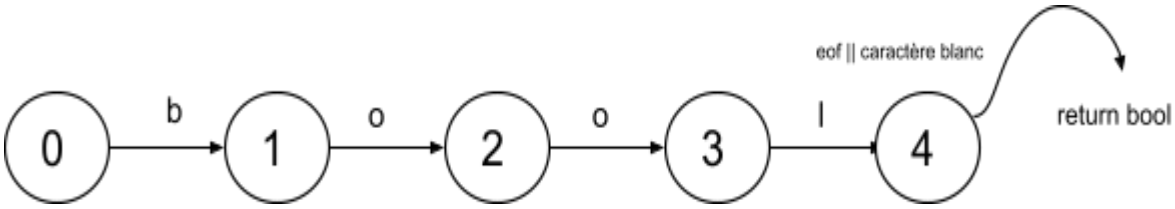
entier :



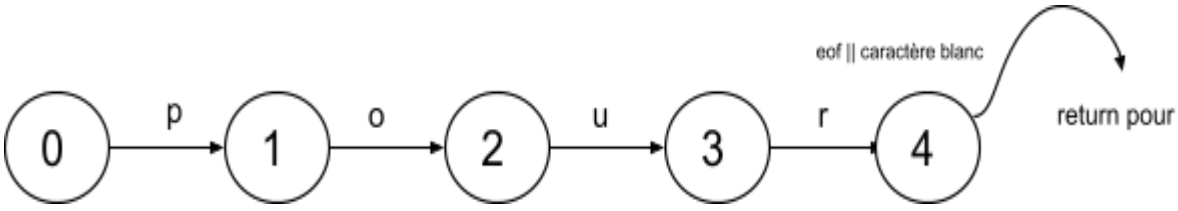
reel :



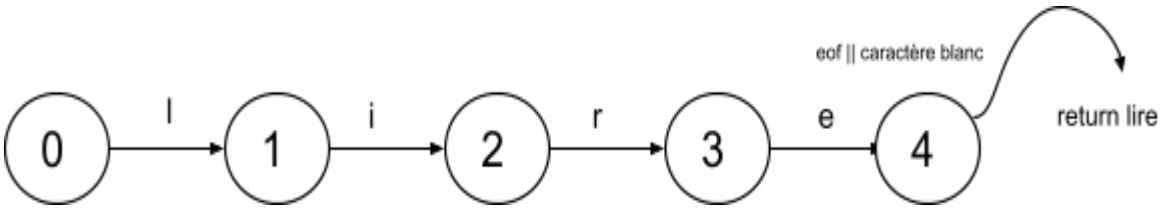
bool :



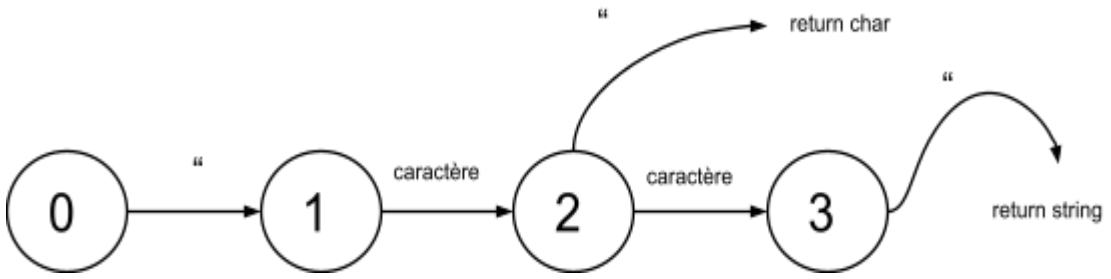
pour :



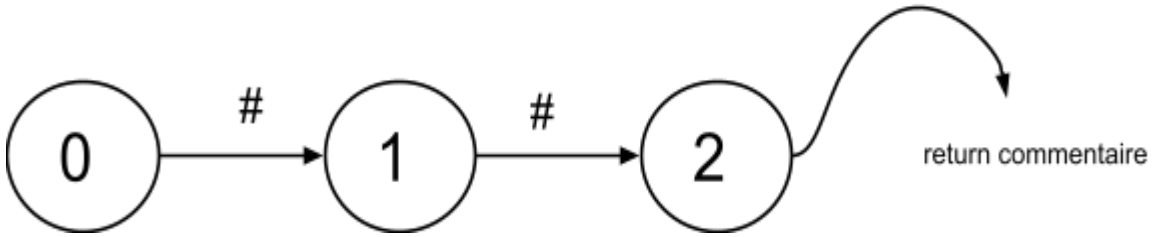
lire :



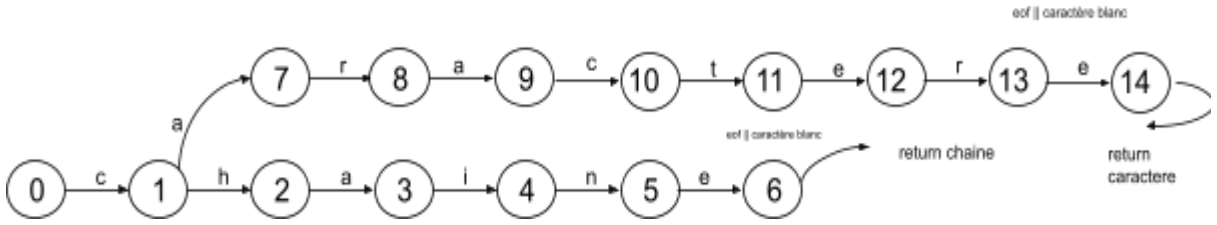
string :



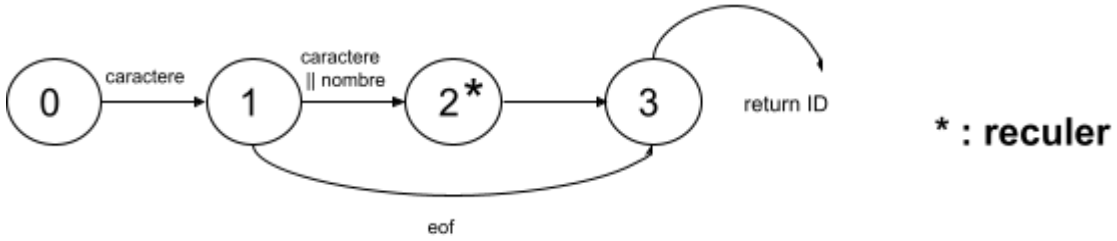
commentaire :



chaine / caractère :

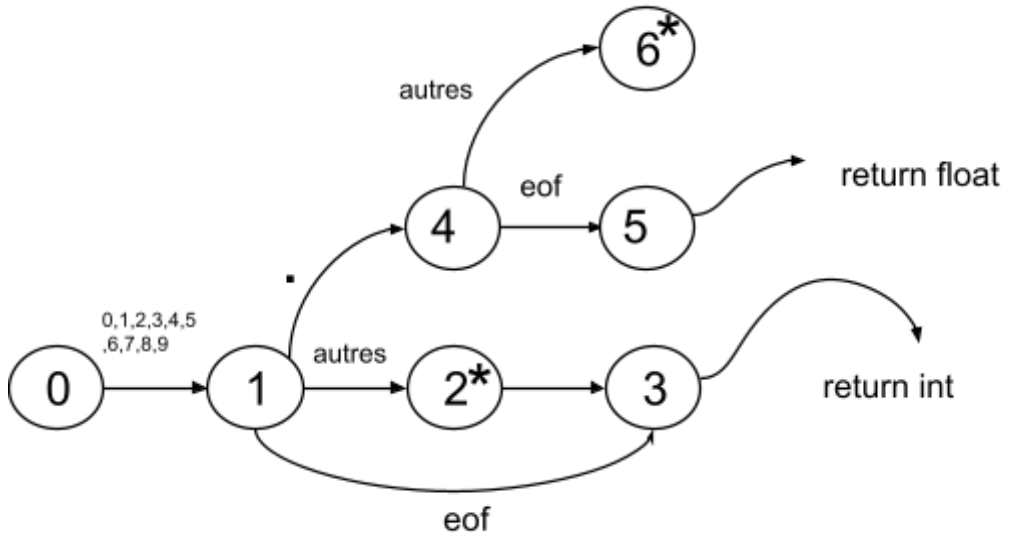


ID :

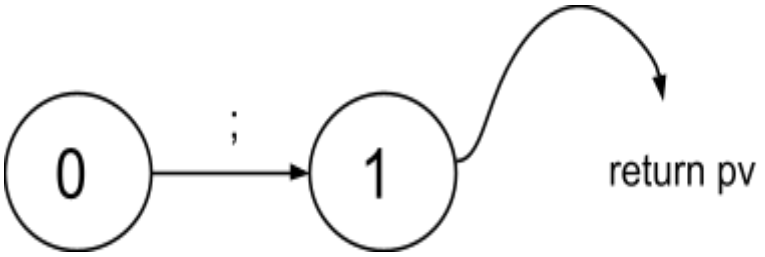


*** : reculer**

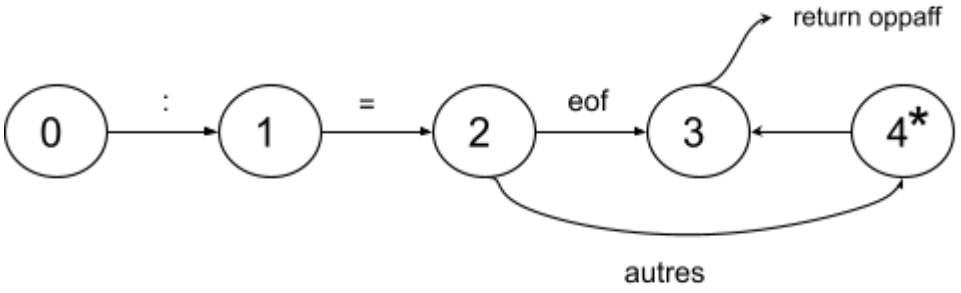
int / float :



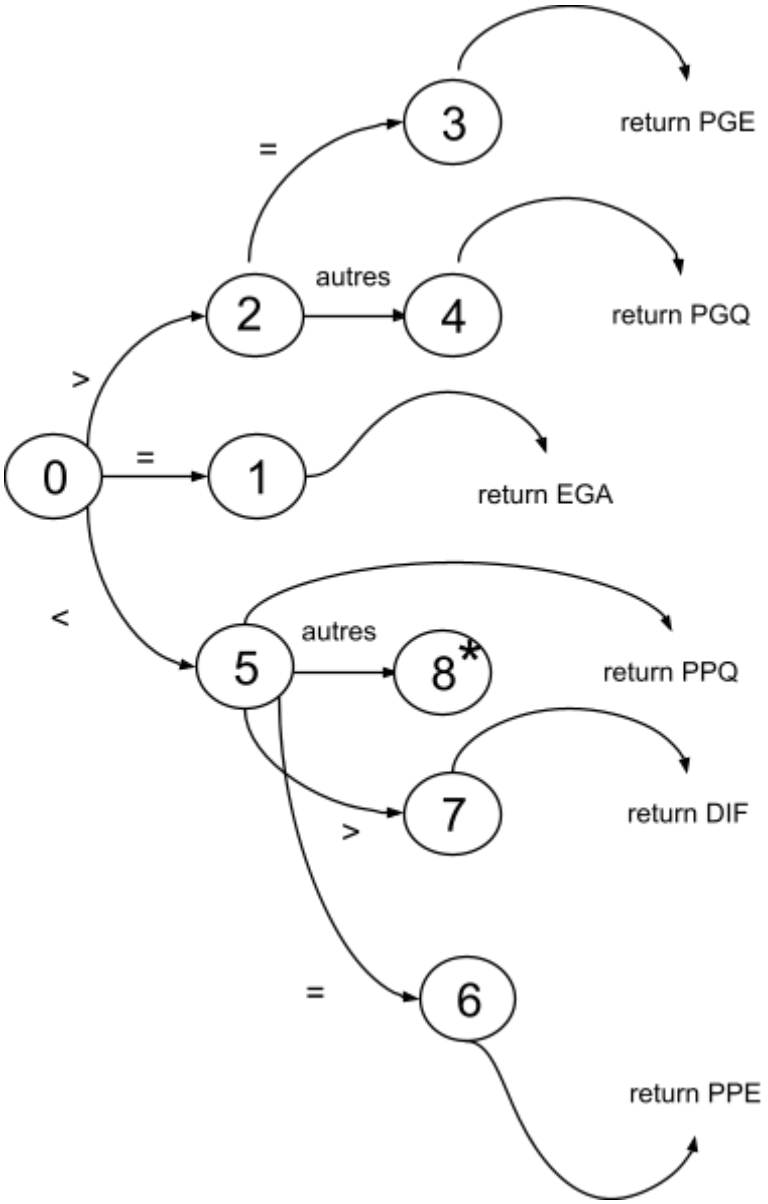
pv :



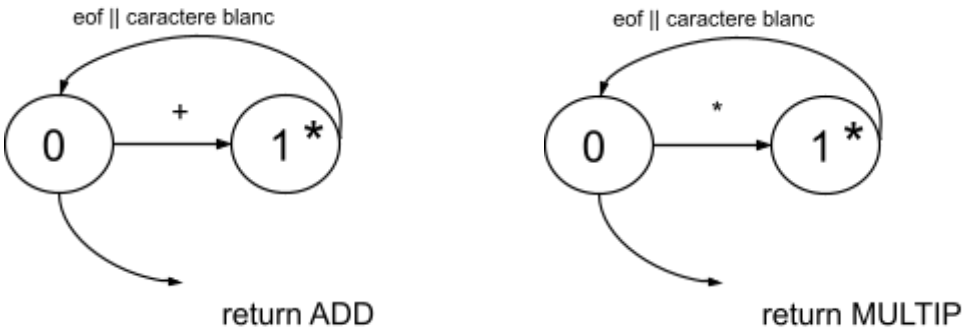
oppaff := :

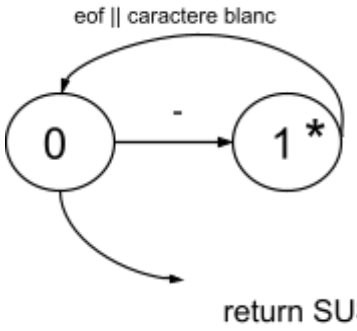


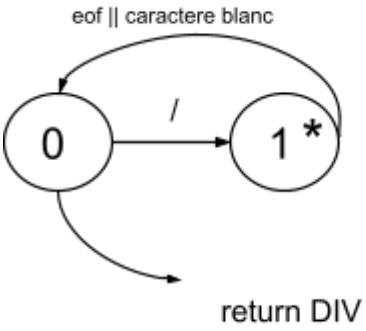
Opprel : < <= > >= <> =



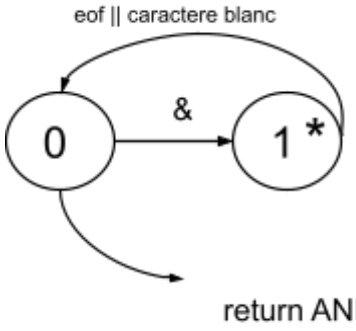
OppArith :

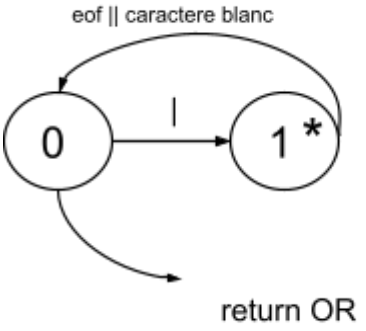


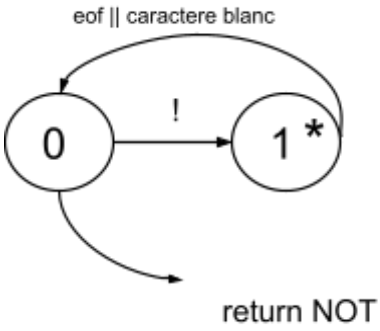

eof || caractere blanc
0
-
1 *
return SUS


eof || caractere blanc
0
/
1 *
return DIV

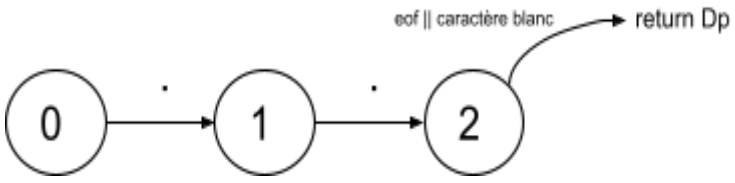
OppLog :


eof || caractere blanc
0
&
1 *
return AND

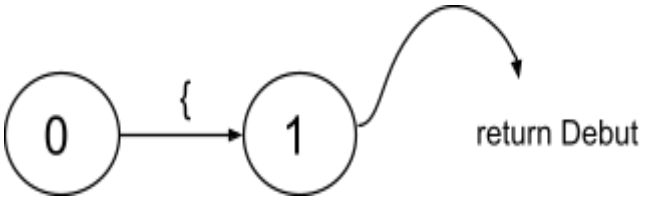

eof || caractere blanc
0
|
1 *
return OR


eof || caractere blanc
0
!
1 *
return NOT

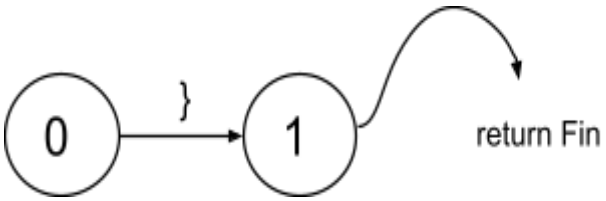
Dp : .. :


eof || caractère blanc
return Dp
0
.
1
.
2

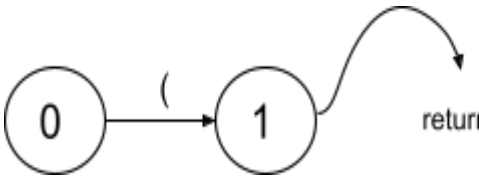
{ :


0
{
1
return Debut

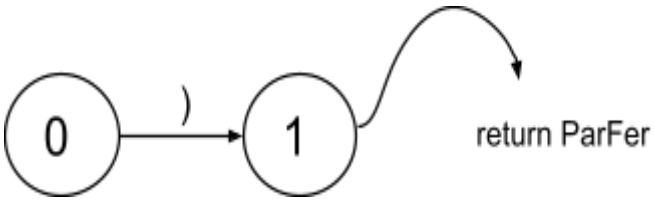
} :


0
}
1
return Fin

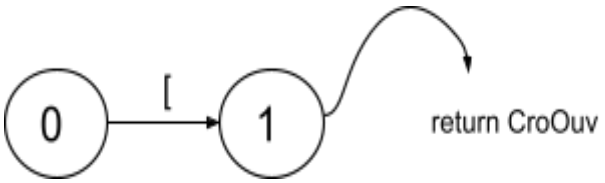
(:


0
(
1
return ParOuv

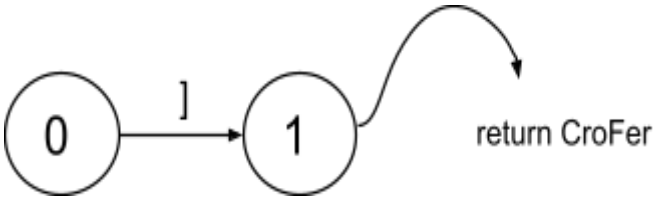
) :


0
)
1
return ParFer

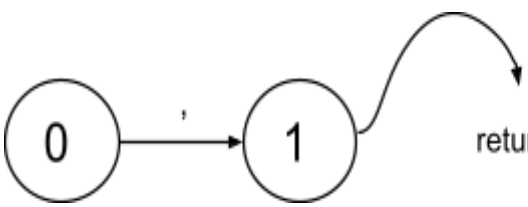
[:


0
[
1
return CroOuv

] :


0
]
1
return CroFer

, :


0
,
1
return v

Grammaire

P -> D { I } P
P -> ''

D -> T K pv D
D -> ''

K -> id
K -> id v K

T -> entier
T -> réel
T -> bool
T -> tab [int] de T
T -> chaine
T -> caractère

I -> id OppAff E pv I
I -> si E alors I finsi pv I
I -> si E alors I sinon I finsi pv I
I -> pour id de entier à entier faire I finpour pv I
I -> lire id pv I
I -> afficher E pv I
I -> ''

O -> OppRel E
O -> OppLog E
O -> OppArith E
O -> ''

E -> (E) O
E -> id O
E -> int O
E -> float O
E -> string O
E -> char O
E -> True O
E -> False