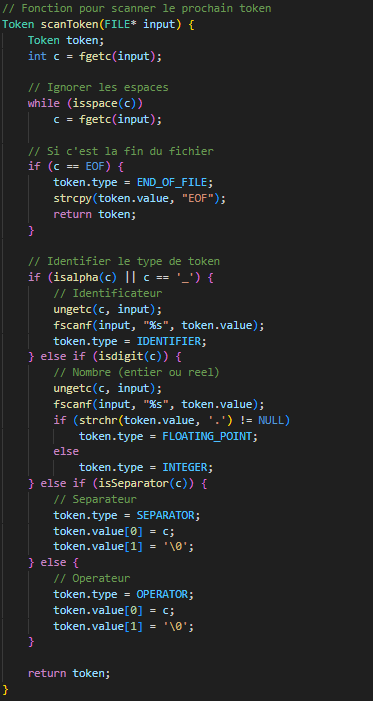
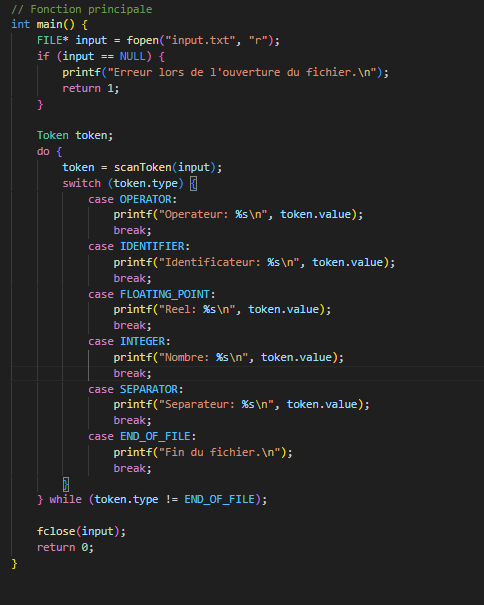
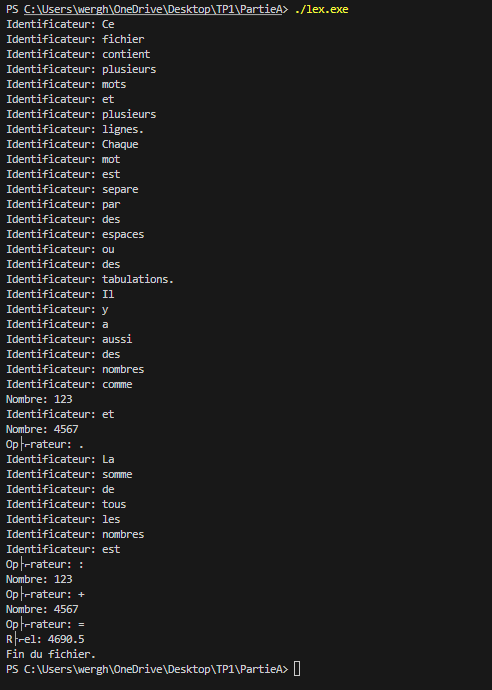
# PARTIE A



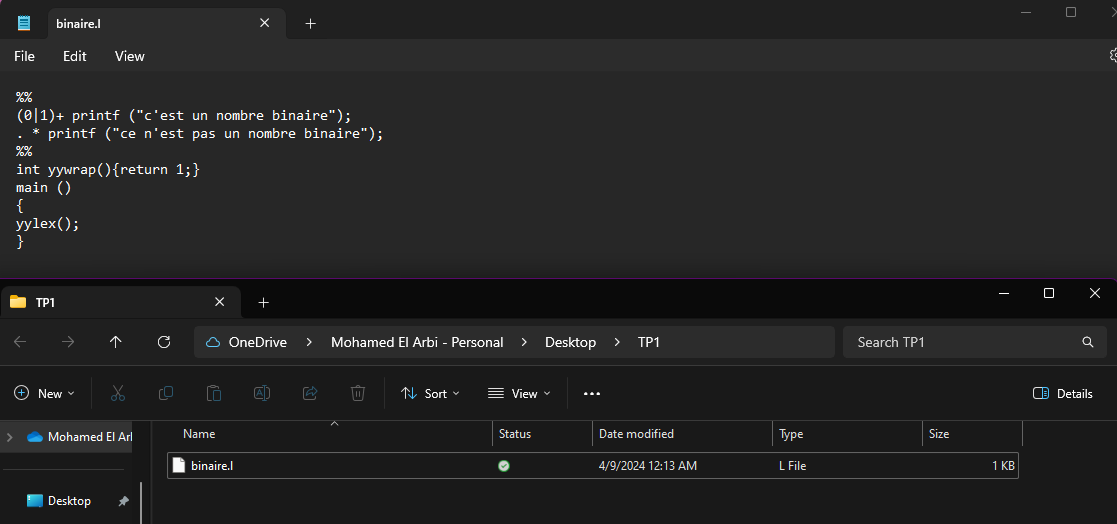


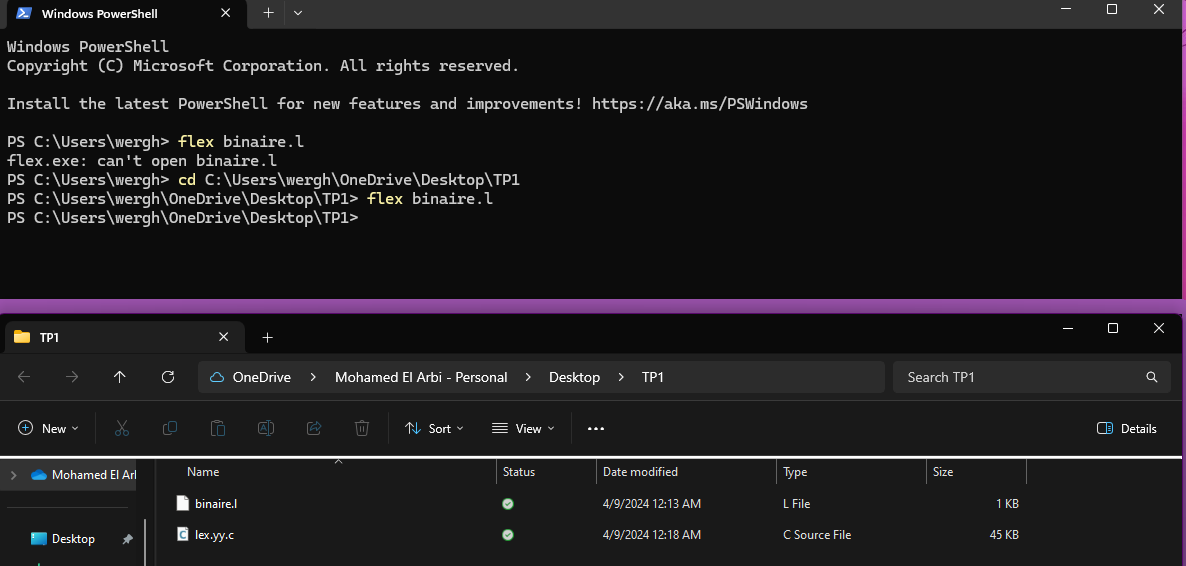


# PARTIE B

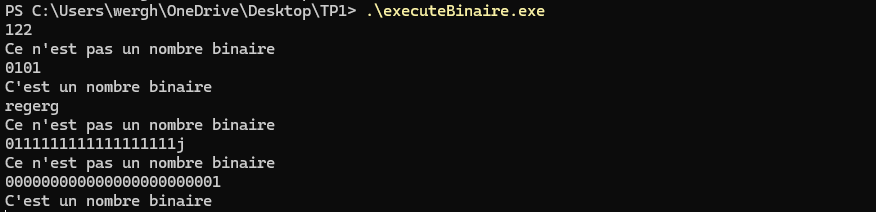
## Partie1

1+2)

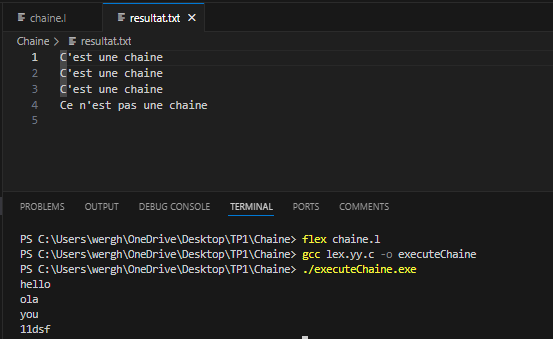




3)

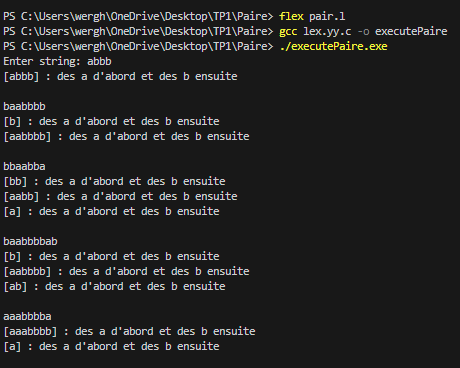


4)

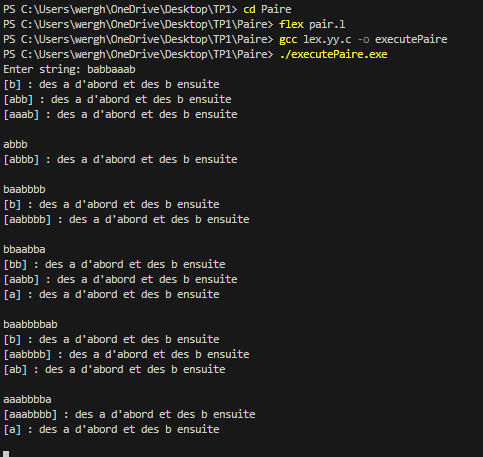


## Partie 2

1+2)

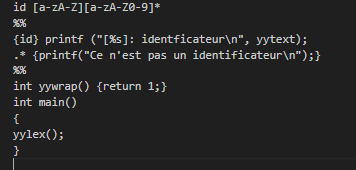


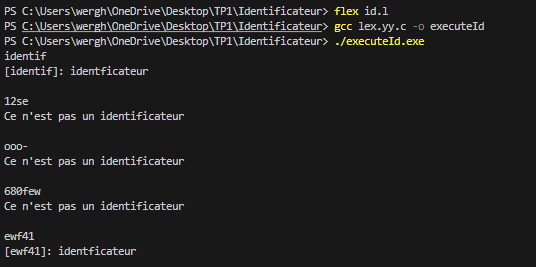
3)



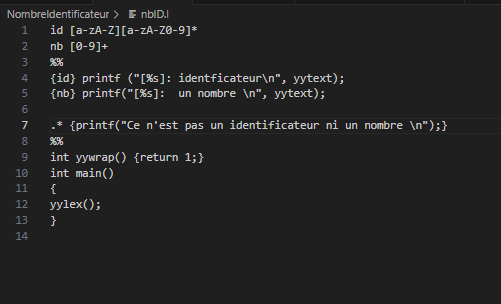
4) on obtient les memes resultats

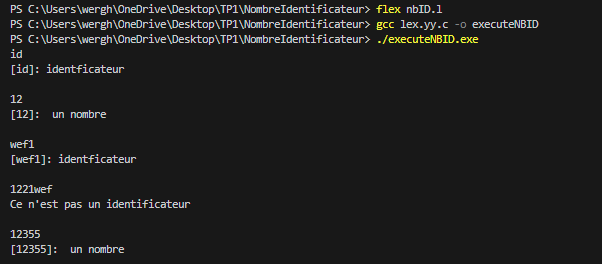
5)





6)





# PARTIE C

## Analyse descendant

NB:le terminal doit être en minuscule

les non-terminaux doivent être en majuscules

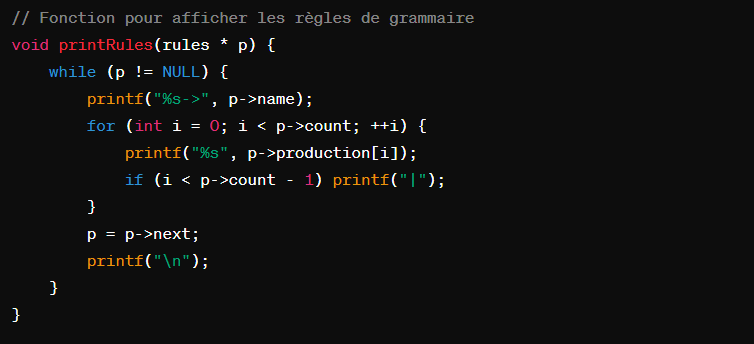
les terminaux et les non-terminaux doivent contenir un caractère au maximum

**les fonctions utilisees**

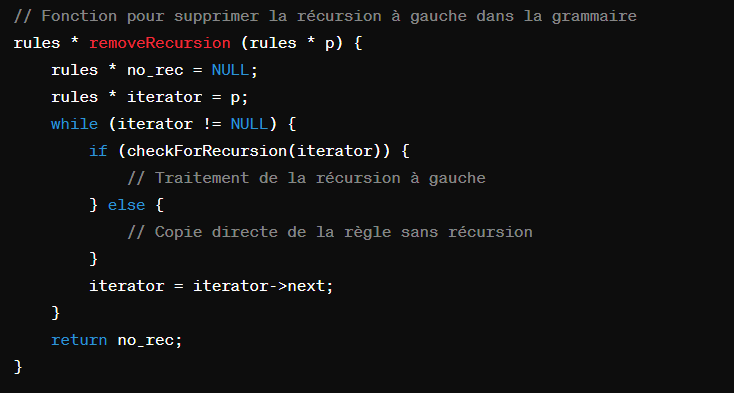
1. saisie() : Cette fonction est responsable de la saisie de la grammaire à partir de l'entrée utilisateur et de la construction de la structure de données correspondante.



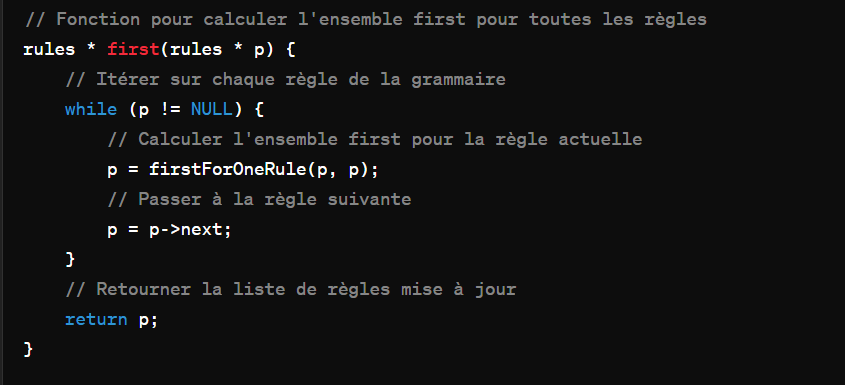
1. printRules() : Cette fonction permet d'afficher les règles de la grammaire pour vérification et débogage.



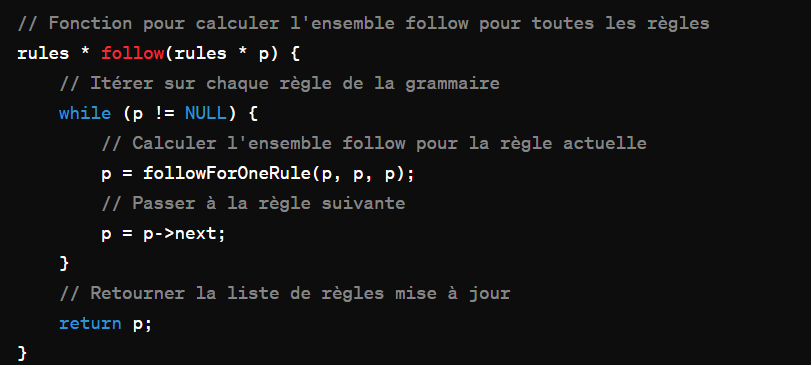
1. removeRecursion() : Fonction pour éliminer la récursion gauche de la grammaire, une étape essentielle dans la préparation de la grammaire pour l'analyse LL(1).



1. first() : Calcul des ensembles first pour chaque symbole non-terminal de la grammaire.



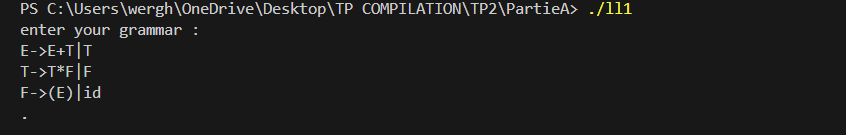
1. follow() : Calcul des ensembles follow pour chaque symbole non-terminal de la grammaire.



1. tableLL1() : Construction de la table d'analyse LL(1) à partir des ensembles first et follow.
2. checkIfvalid() : Cette fonction utilise la table d'analyse LL(1) pour vérifier si une chaîne donnée est acceptée par la grammaire.

**Exemple:**

Ecrire notre grammaire:



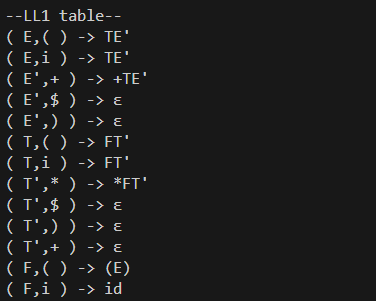
Éliminer la récursivité a gauche:



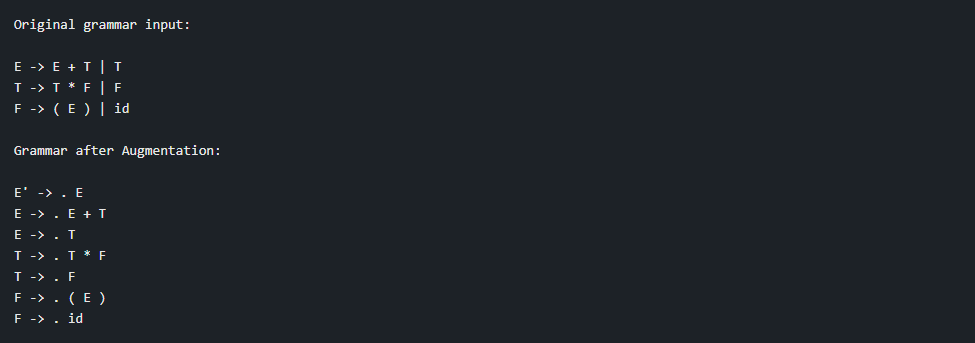
Deduire le premier et le suivant de chaque règle:

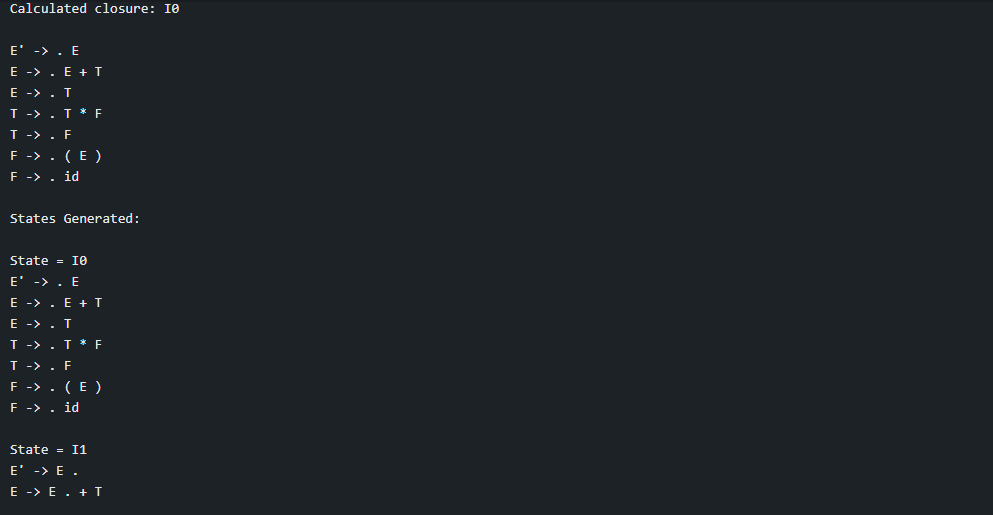


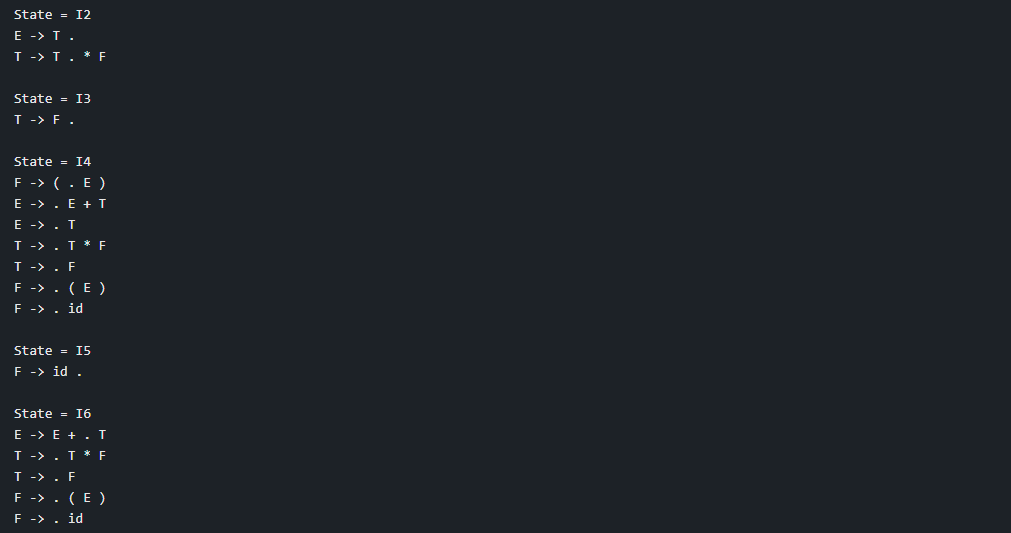
Creation du table LL1: verification:

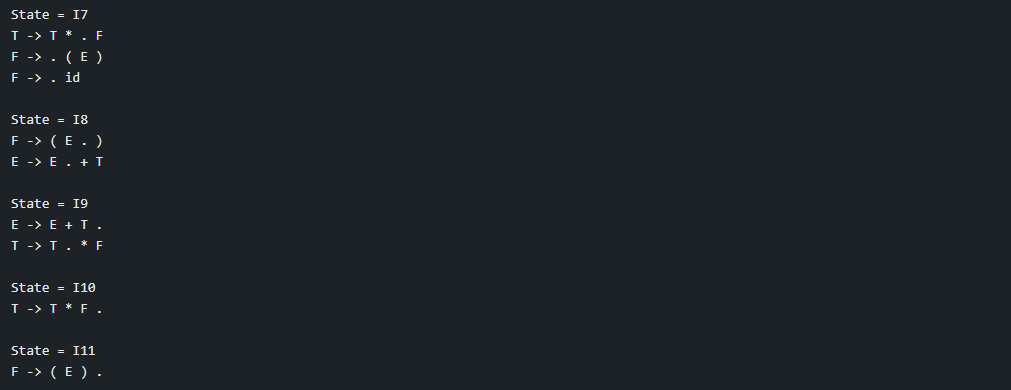


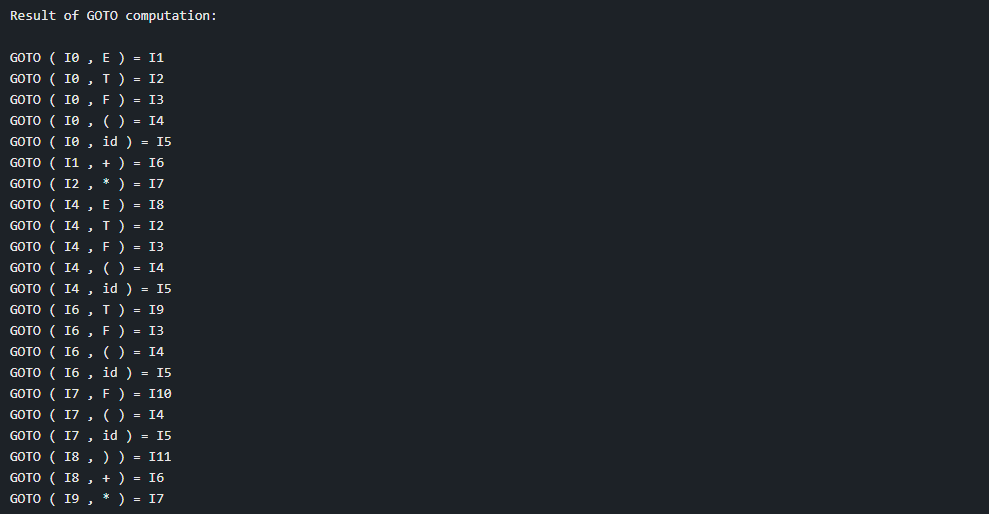
## Analyse ascendant

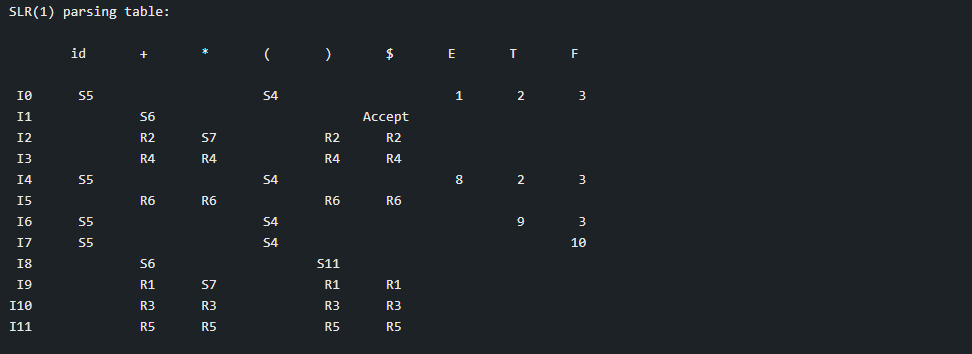






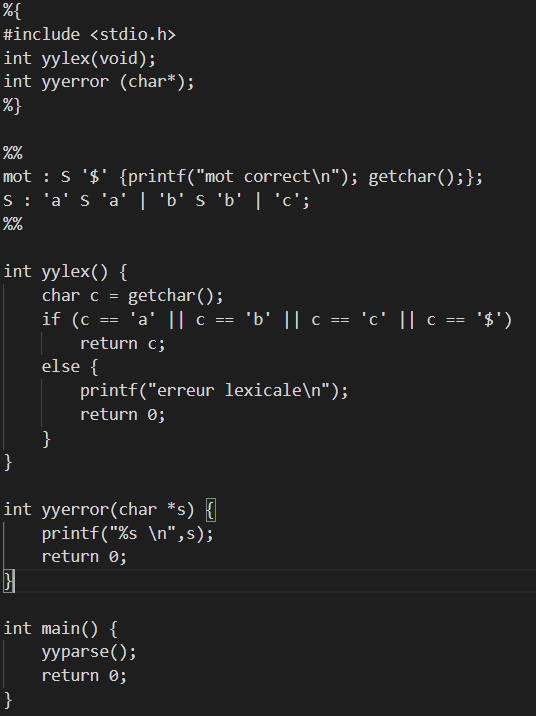


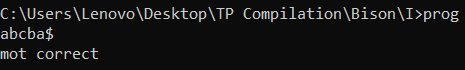
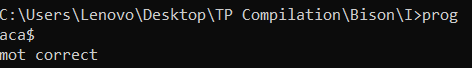




# Partie D

## Question II

****

****

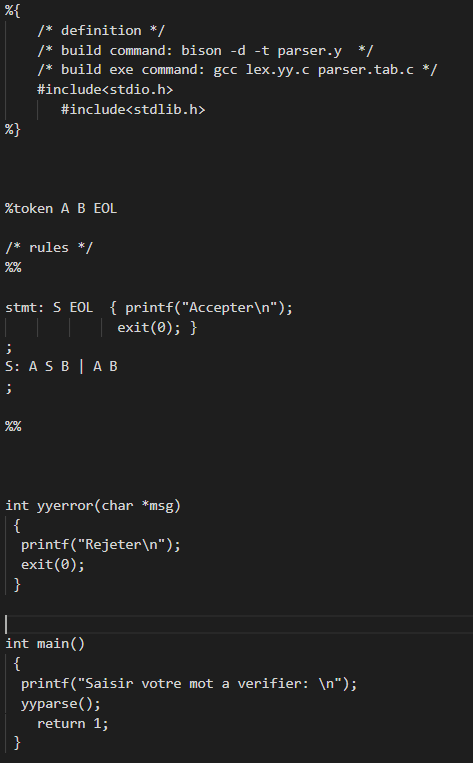
les mots accepté sont les mot du grammaire :

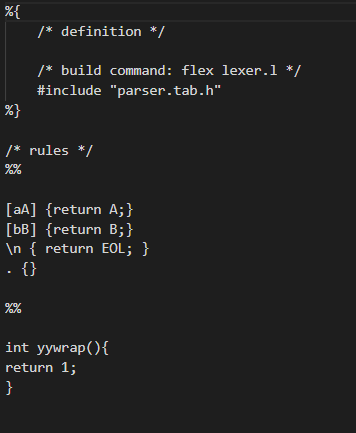
S->aSa | bSb | c

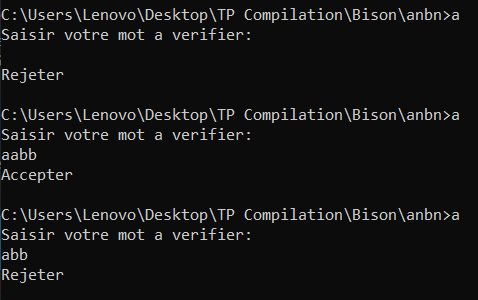
et à la fin on ajoute $

## Question III

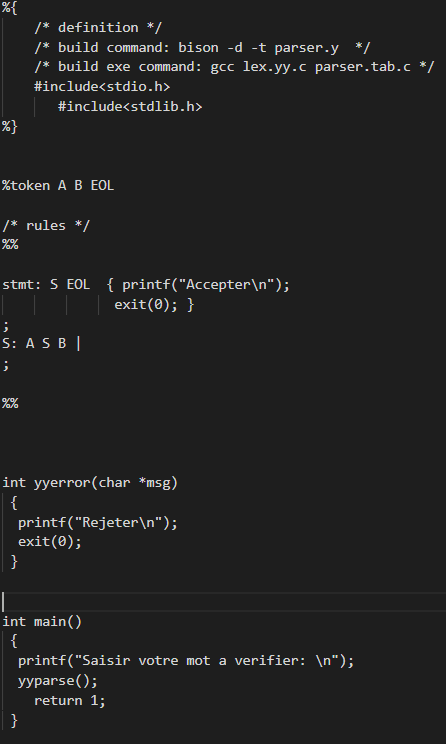
1. L={anbn, n>0}

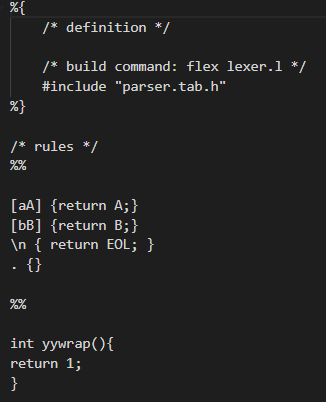


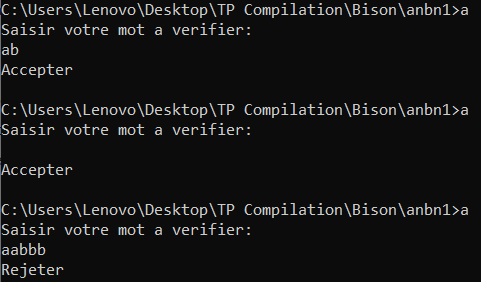




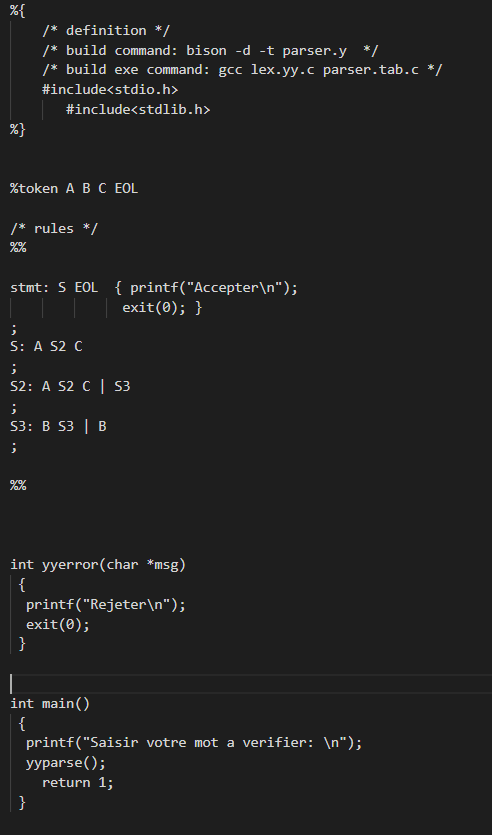
1. L={anbn, n>=0}

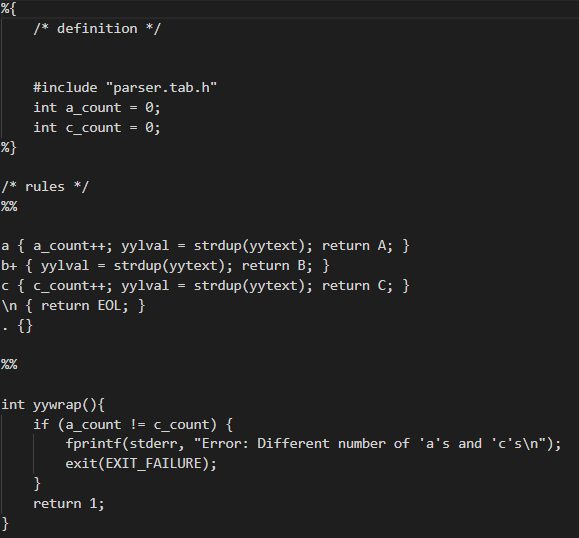


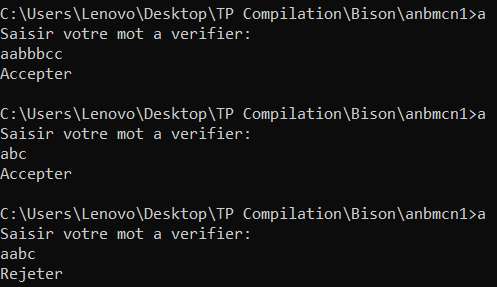




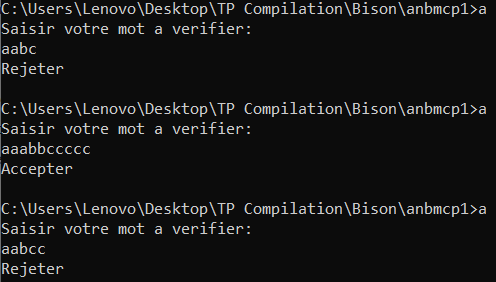
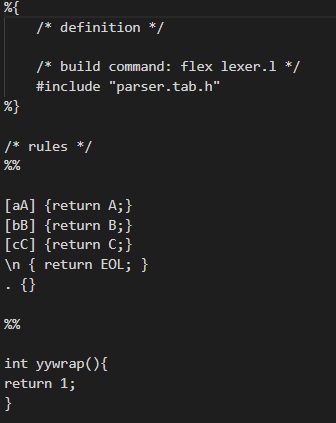
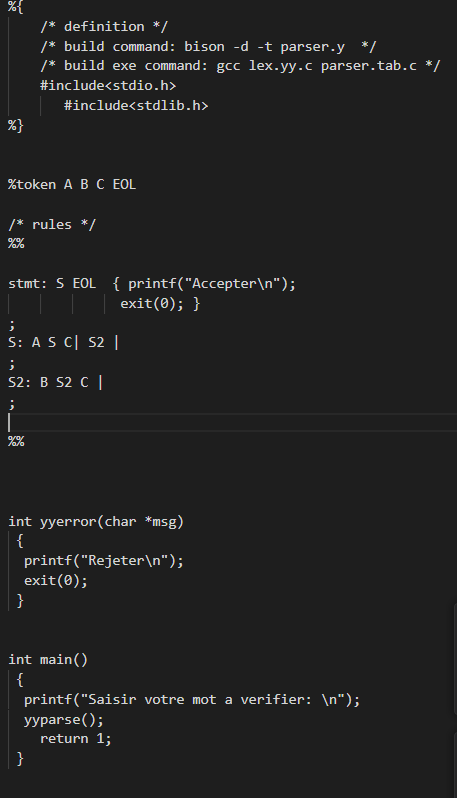
1. L={anbmcn \ n,m>=1}





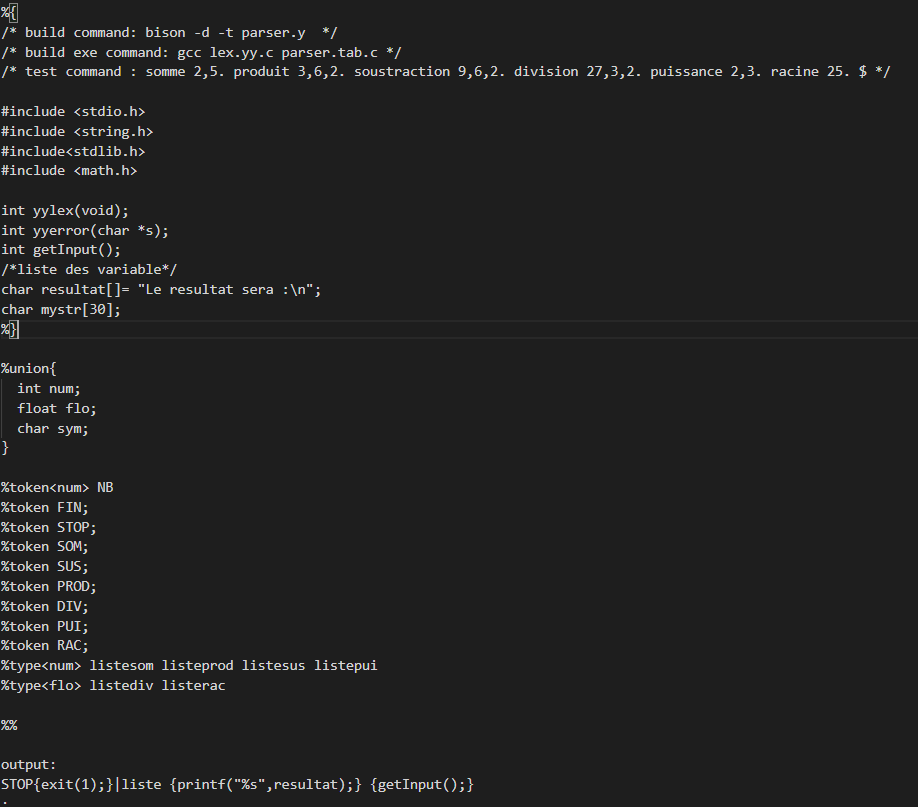


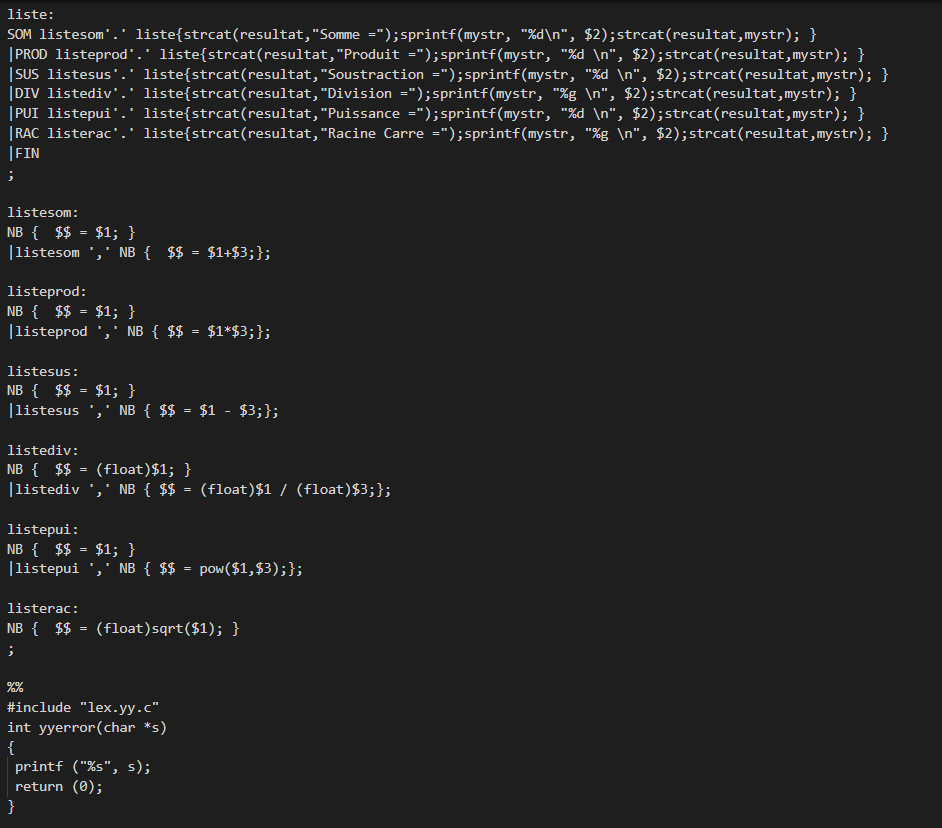
1. L={anbmcp \ n+m=p}

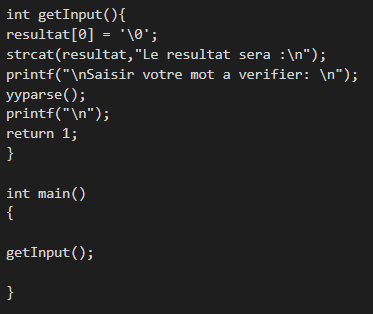


## Question IV

parser.y







lexer.L:

