**Contrôle**

**Le : 15/01/2023 Durée : 1h30m**

**Exercice 01 ( 04 pts) :** Soit le code Lex suivant :

*%%  
aa { printf("1"); }  
b?a+b? { printf("2"); }  
b?a\*b? { printf("3"); }  
.|\n { printf("4"); }*

1. Donnez un exemple d'une entrée de ce fichier qui produira 123 en sortie, ou expliquez pourquoi il n'en existe pas.
2. Donnez un exemple d'entrée à ce fichier qui produira 321 comme sortie, ou expliquez pourquoi il n'en existe pas.

**Exercice 02 ( 08 pts) :** Soit la grammaire sur l’alphabet {a b ( ) + \*}

*R→ RR|R+R | R\* | (R)* | a | b avec + l’opérateur binaire (ou)

1. Montrer que cette grammaire est ambigüe.
2. Construire une grammaire équivalente de type LL(1), avec les règles de priorité suivantes :

* L’opérateur de répétition \* est le plus prioritaire et est associatif à gauche
* L’opérateur de concaténation a la seconde plus haute priorité, il est  
  associatif à gauche
* L’opérateur d’alternative + (c’est le ou |) est le moins prioritaire et est associatif à gauche.

1. Construire la table d’analyse pour la nouvelle grammaire.

**Exercice 03 ( 08 pts) :** Soit La grammaire suivante :

*S→cS | T*

*T→aCa | bCb | aDb | bDa*

*C→z*

*D→z*

1. Donnez l’automate LR(1) de la grammaire G :
2. Cette grammaire est-elle LR(0) ? Est-elle LALR(1)? justifiant vos réponses
3. Donnez la table de l’analyseur LR(1) pour cette grammaire, et simulez en le fonctionnement sur la chaîne *cccazb.*

**Corrigé type**

**Fait le : 15/01/2023 Durée : 1h30**

**Exercice 01 ( 04 pts) :** Soit le code Lex suivant :

*%%  
aa { printf("1"); }  
b?a+b? { printf("2"); }  
b?a\*b? { printf("3"); }  
.|\n { printf("4"); }*

1. Donnez un exemple d'une entrée de ce fichier qui produira **123** en sortie, ou expliquez pourquoi il n'existe pas **( 02 pts)**.

*Aucune entrée de ce type n'existe. Pour que l'entrée produise 1 au début, les deux premiers caractères doivent être "aa". Considérons maintenant le personnage suivant. Si c'est un 'a', alors, en utilisant la chaine maximale, le scanner correspondrait à la règle 2 sur la règle 1, donc 1 ne serait pas imprimé. Si c'est un 'b', c'est la même chose. Si c'est autre chose, alors la règle 1 correspondra à "aa", mais le caractère suivant ne correspondra à aucune des règles précédentes, donc un 4 sera imprimé au lieu d'un 2.*

1. Donnez un exemple d'entrée à ce fichier qui produira **321** comme sortie ; Une seule chaine qui est : **bbbabaa ( 02 pts)**

**Exercice 02 ( 08 pts) :** Soit la grammaire sur l’alphabet {a b ( ) + \*}

*R→ RR|R+R | R\* | (R)* | a | b avec + l’opérateur binaire (ou)

1. Montrer que cette grammaire est ambigüe : Deux arbres différents pour la même chaine **(01 pt).**
2. Construire une grammaire équivalente de type LL(1), en respectant les règles de priorité: **(01.5 pt)**

R→R+T|TF

T→TF|ε

F→Z\*|Z

Z→(R)|a|b

1. Cette grammaire contient deux cas de récursivités à gauche et un cas de factorisation, il faut les éliminer :

R→R+T|TF 🡺 R→TFR’

R→+TR’|ε

T→TF|a|b 🡺 T→FT|ε

F→F\*|Z 🡺 F→ZF’

F’→\*F’|ε

Alors la nouvelle grammaire G’ est comme suit **(01.5 pts):** :

R→TFR’

|  |  |  |
| --- | --- | --- |
| **(02 pts):** | Premiers | Suivants |
| R | ( a b | $ ) |
| R’ | + ε | $ ) |
| T | ( a b ε | + $ ) |
| F | ( a b | ( a b + $ ) |
| F’ | \* ε | ( a b + $ ) |
| Z | ( a b | ( a b \* + $ ) |

R’→+TR’|ε

T→FT|ε

F→ZF’

F’→\*F’|ε

Z→(R)|a|b

1. La table d’analyse pour la nouvelle grammaire **(02 pts).**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | a | b | ( | ) | + | \* | $ |
| R | TFR’ | TFR’ | TFR’ |  |  |  |  |
| R’ |  |  |  | ε | +TR |  | ε |
| T | FT | FT | FT| | ε | ε |  | ε |
| F | ZF’ | ZF’ | ZF’ |  |  |  |  |
| F’ | ε | ε | ε | ε | ε | \* F’ | ε |
| Z | a | b | (R)| |  |  |  |  |

**Exercice 03 ( 08 pts) :**

**Contrôle**

**le : 25/01/2023 Durée : 1h30**

**Exercice 01 (04 pts)**

Sur l'alphabet {*a , b, c}*, donner, lorsque c'est possible, une expression régulière pour chacun des langages suivants :

1. Ensemble des mots w ne contenant trois occurrences successives de *a* ;
2. Chaînes dans lesquelles chaque *a* est immédiatement suivi d'au moins deux *b*;
3. Chaînes dans lequel aucun *a* n'apparaît après le premier *b* (s'il en existe un)
4. Chaînes qui commencent et se terminent par des symboles différents.

**Exercice 02 (08 pts)**

Soit la grammaire G :

S → bAb | bBa

A → aS | CB

B → b | BC

C → c | cC

1. Montrer que cette grammaire est ambigüe.
2. Transformez-la pour qu’elle devienne LL(1) ?
3. Construire la table d’analyse LL(1) de la nouvelle grammaire?
4. Analyser la chaine « babccca$».

**Exercice 03 (08 pts)** Soit la grammaire G :

S ⟶ Aa | SaSb   
A ⟶Ba | Sb | b  
B ⟶aBb | Bba | a

1. Donnez l’automate LR(1) de la grammaire G.
2. Construire la table LR(1) de G? G est-elle LR(1) ? justifiant vos réponses
3. G est-elle LALR(1) ? justifiant vos réponses

**Contrôle**

**le : 25/01/2023 Durée : 1h30**

**Exercice 01 (04 pts)**

Sur l'alphabet {*a , b, c}*, donner, lorsque c'est possible, une expression régulière pour chacun des langages suivants :

1. Ensemble des mots w ne contenant trois occurrences successives de *a* ;
2. Chaînes dans lesquelles chaque *a* est immédiatement suivi d'au moins deux *b*;
3. Chaînes dans lequel aucun *a* n'apparaît après le premier *b* (s'il en existe un)
4. Chaînes qui commencent et se terminent par des symboles différents.

**Exercice 02 (08 pts)**

Soit la grammaire G :

S → bAb | bBa

A → aS | CB

B → b | BC

C → c | cC

1. Montrer que cette grammaire est ambigüe.
2. Transformez-la pour qu’elle devienne LL(1) ?
3. Construire la table d’analyse LL(1) de la nouvelle grammaire?
4. Analyser la chaine « babccca$».

**Exercice 03 (08 pts)** Soit la grammaire G :

S ⟶ Aa | SaSb   
A ⟶Ba | Sb | b  
B ⟶aBb | Bba | a

1. Donnez l’automate LR(1) de la grammaire G.
2. Construire la table LR(1) de G? G est-elle LR(1) ? justifiant vos réponses
3. G est-elle LALR(1) ? justifiant vos réponses

**Corrigé type**

**Fait le : 15/01/2023 Durée : 1h30**

**Exercice 01: (04 pts)**

**Exercice 02: (08 pts)**

1. La grammaire est ambiguë (la chaîne bbcca a deux dérivations les plus à gauche 1:

S *→* bBa *→* bBCa *→* bbCa *→* bbcCa *→* bbcca,

S *→* bBa *→* bBCa *→* bBCCa *→* bbCCa *→* bbcCa *→* bbcca.

1. Transformation de la grammaire pour qu’elle devienne LL(1)

S → bAb | bBa

A → aS | CB

B → b | BC

C → c | cC

La factorisation à gauche des non-terminaux S et C est simple. Le non-terminal B était la source à la fois des problèmes de récursivité à gauche et d'ambiguïté, et les deux problèmes sont résolus en générant un seul b, puis zéro ou plusieurs c (via un nouveau non-terminal D). La nouvelle grammaire sera comme suit 2 :

S→bT

T→Ab| Ba

A →aS| CB

B→bD

C→cD

D→cD| ε

1. Construction de la table d’analyse LL(1) de la nouvelle grammaire?.

|  |  |  |
| --- | --- | --- |
| 2 | Premiers | suivant |
| S | b | $,b |
| T | a, b,c | $,b |
| A | a,c, | b |
| B | b | a,b, |
| C | c | b |
| D | c, ε | a,b, |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1.5 | a | b | c | $ |
| S |  | bT |  |  |
| T | Ab | Ba | Ab |  |
| A | aS |  | CB |  |
| B |  | bD |  |  |
| C |  |  | cD |  |
| D | ε | ε | cD |  |

1. Analyser la chaine « babccca$».1.5

**Rattrapage**

**Le : 30/01/2023 Durée : 1h30**

**Exercice 01 (03 pts) :** Considérons l’alphabet Σ = {a, b, c}. Pour chacun des langages suivants sur Σ, donner une expression régulière qui dénote :

1. Les chaînes dont Il n'y a jamais deux b à la suite.
2. Les chaînes qui commencent par a et ont une longueur paire.
3. Les mots dans lesquels la chaine *ab*, s’il apparaît, est suivi de *ccc* ;

**Exercice 02 (03 pts) :**

Ecrire un programme Lex qui compte le nombre d’occurrence de chaque lettre minuscule dans un fichier texte. Le résultat doit être stocké dans un tableau tab de 26 entiers: on met dans tab[0] le nombre d’occurrence de ‘a’ , …, et ainsi de suite.

**Exercice 03 (06 pts) :** Considérons la grammaire des expressions post-fixe :

E → EE+| EE\*| nb

1. Eliminer la récursivité à gauche.
2. Factoriser à gauche.
3. Calculer les ensembles Premier et Suivant de la nouvelle grammaire.
4. La nouvelle grammaire est-elle LL(1) ? Sinon, la rendre LL(1).

**Exercice 4(06 pts) :** Soit la grammaire *G*2 :

*S → A* a *|* a

*A → B* c

*B → S* b *| B* b *| ε*

1. Donnez l’automate LR(1) de la grammaire G.
2. Construire la table LR(1) de G? G est-elle LR(1) ? justifiant vos réponses
3. G est-elle LALR(1) ? justifiant vos réponses

**Corrigé type de Rattrapage**

**Le : 29/03/2022 Durée : 1h**

**Exercice 01 (01+01+01 pts) :** Considérons l’alphabet Σ = {a, b, c}. Pour chacun des langages suivants sur Σ, donner une expression régulière qui dénote :

1. Les chaînes dont Il n'y a jamais deux b à la suite.

(a|b(a|c)|c)\*|(b|ε)

1. Les chaînes qui commencent par a et ont une longueur paire.

(a(a|b|c))( (a|b|c) (a|b|c))\*

1. Les mots dans lesquels la chaine *ab*, s’il apparaît, est suivi de *ccc* ;

b\*(a|cb\*|bccc)\*

**Exercice 02 (03 pts) :**

%{

#include<stdio.h>

Int tab[26] ;

%}

%%

[a-z] {tab[‘a’-yytext[0]]++ ;}

.|\n {}

%%

void main()

{

int i=0 ;

for(i=0 ;i<26 ;tab[i++}=0) ;

yylex() ;

for(i=0 ;i<26 ;i++) printf(“%c=%d”, i, tab[i}) ;

}

**Exercice 03 (06 pts) :** Considérons la grammaire des expressions post-fixe:

E → EE+| EE\*| nb

1. Eliminer la récursivité à gauche **(01 pt)**.

E → nbE’

E → E+E’| E\*E’|ε

1. Factoriser à gauche **02pts**.

E → E+E’| E\*E’|ε

E → E(+E’| \*E’)|ε

E → EE’’|ε

E’’ → +E’| \*E’

1. Calculer les ensembles Premier et Suivant de la nouvelle grammaire.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **01.5 pts**. | nb | + | \* | $ |
| E | nbE’ |  |  |  |
| E’ | EE’’ | ε | ε | ε |
| E’’ |  | +E’ | \*E’ |  |

|  |  |  |
| --- | --- | --- |
| **02 pts** | Premiers | Suivants |
| E | nb | + \* $ |
| E’ | nb ε | + \* $ |
| E’’ | + \* | + \* $ |

1. Oui cette grammaire est LL(1) parce qu’Il n’existe pas des conflits dans la table d’analyse (**0.5 pts**.)

**Exercice 4(06 pts) :** Soit la grammaire *G*2 :

*S → A* a *|* a

*A → B* c

*B → S* b *| B* b *| ε*

1. Donnez l’automate LR(1) de la grammaire G.
2. Construire la table LR(1) de G? G est-elle LR(1) ? justifiant vos réponses
3. G est-elle LALR(1) ? justifiant vos réponses