Université Mohamed Khider – Biskra
Faculté des Sciences Exactes et des
Sciences de la Nature et de la Vie
Département d'Informatique

Module: **Compilation**
Niveau : **3ème Licence**
Année universitaire : **2017/2018**
Documents non autorisés

Contrôle

Date : 10/01/2018 **Durée : 1h30m**

Exercice 01 (04 pts) :

Soit l'alphabet A = {a,b}, définir les expressions régulières pour les langages:

1. Tous les mots non vides contenant un nombre impair de *b.*
2. Tous les mots non vides contenant un nombre pair de *a*.
3. Tous les mots non vides ne contenant pas deux *a* consécutifs.
4. Tous les mots ne contenant pas plus que deux *a* consécutifs.

Exercice 02 (04 pts)

Les commentaires de documentation en Java, sont encadrés par les motifs /** et */, ils peuvent se terminer par une liste de motifs spéciaux, appelés tags. Le nom d'un tag commence par le caractère arobase (@) et figure nécessairement en début de ligne (en ignorant les espaces et les *). Si des mots commençant par @ figurent ailleurs qu'en début de ligne, ils ne seront pas considérés comme des tags. Chaque tag a un texte associé (éventuellement sur plusieurs lignes) qui se termine au tag suivant ou au motif final */.

Exemple :

```
/**
* Returns the character at the specified index.
*
* @param index the index of the desired character.
* @return the desired character.
* @exception StringIndexOutOfRangeException
*                if the index is not in the range <code>0</code>
*               to <code>length()-1</code>.
*/
```

Ecrire un analyseur lexical, en lex, qui **affiche dans un fichier** nommé *docfile* les tags @param, @return et @exception des commentaires de documentation figurant dans le fichier java, avec leur texte associé.

Exercice 03 (12 pts)

Partie 1: On considère la grammaire G définie par :

$S \longrightarrow S \vee S \mid S \wedge S \mid \neg S \mid (S) \mid b$

1. Cette grammaire n'est pas LL(1) : pourquoi ?
2. Donner une grammaire G1 équivalente à G et qui vous semble LL(1).
3. Simuler le fonctionnement de l'analyseur LL(1) de cette grammaire sur le mot :
   $\neg$ (b $\vee$ b) $\wedge$ b.

Partie 2: Soit la grammaire G suivante, écrite dans le format reconnu par yacc:

```
%token b
%%
S⟶ S'V' S
   | S ' ∧ ' S
   | b
   ;
%%
```

Le token b représente une variable booléenne.

1. Construisez l'automate LR de cette grammaire
2. Il est évident que cette n'est pas de type LR(0), Quelle sorte de conflits apparaissent ?
3. Proposez une solution pour résoudre ces conflits. Indiquer les modifications à apporter dans le fichier yacc.

Université Mohamed Khider – Biskra
Faculté des Sciences Exactes et des
Sciences de la Nature et de la Vie
Département d'Informatique

Module: **Compilation**
Niveau : **3ème Licence**
Année universitaire : **2017/2018**
Documents non autorisés

Corrigé type

Fait le : 10/01/2018 **Durée : 1h30m**

Exercice 01 (04 pts)

Soit l'alphabet A = {a,b}, définir les expressions régulières pour les langages :

1. Tous les mots non vides contenant un nombre impair de *b* ; a*b(a*ba*b)* a*
2. Tous les mots non vides contenant un nombre pair de *a*; (b*ab*a) (b*ab*a)* b*|bb*
3. Tous les mots ne contenant pas deux *a* consécutifs : (ab|b)*(a|b)
4. Tous les mots ne contenant pas plus que deux *a* consécutifs ; (ab|b)*(a|b) (ab|b)*a| ε

Exercice 02 (04 pts)

```
%{
#include <stdio.h>
#include <stdlib.h>
FILE *yyin, *yyout ;
%}
%start COM TAG
Tags |
%%
<INITIAL>"/**" {BEGIN(COM);}
<COM> ^(\*|' ')*« @param »{ BEGIN(TAG)}; fprintf(yyout, « %s », « @param »);}
<COM> ^(\*|' ')* « @return »{ BEGIN(TAG)}; fprintf(yyout, « %s », « @return »);}
<COM> ^(\*|' ')* « @exception»{ BEGIN(TAG)}; fprintf(yyout, « %s », « @exception»);}
<COM>"*/" {BEGIN(INITIAL)}
<COM> . {}
<TAG> ^(\*|' ')*« @param »{fprintf(yyout, « %s », « \n @param »);}
<TAG> ^(\*|' ')* « @return »{ fprintf(yyout, « %s », « \n@return »);}
<TAG> ^(\*|' ')* « @exception»{fprintf(yyout,« %s », « \n @exception»);}
<TAG> "*/" {BEGIN(INITIAL)}
<TAG> (.|'\n') { fprintf(yyout, « %c », yytext[0]);}
%%
int yywrap(){ return 1; }

int main(int argc,char **argv){

  yyin = fopen(argv[1],"r");
  yyout = fopen("docfile","w");
  yylex();
fclose(yyin) ;
fclose(yyout) ;

}
```

Exercice 03 (12 pts)

Partie 1 : On considère la grammaire G définie par :

$S \longrightarrow S \vee S \mid S \wedge S \mid \neg S \mid (S) \mid b$

1. Cette grammaire ambigüe puisqu'elle ne respecte pas la priorité et l'associativité des opérateurs booléens **(0.5pt).**
2. La grammaire G1 équivalente à G et qui vous semble LL(1).
   Il faut d'abord enlever l'ambigüité de la grammaire **(01.5 pt).**:

X⟶ X ∨ Y| Y

Y⟶ Y ∧ Z |Z

Z⟶ ¬ T | T

T⟶ (X) | b

Il faut donc éliminer la récursivité à gauche **(01.5 pt).**

X⟶ YX'

X'⟶ ∨ YX'| ε

Y⟶ ZY'

Y'⟶ ∧ ZY' | ε

Z⟶ ¬ T | T

T⟶ (X) | b

3. Construction de la table d'analyse LL(1) :

(01.5 pt)	Premiers	Suivants
X	¬ , (, b	$,)
X'	∨ , ε	$,)
Y	¬ , (, b	$,) , ∨
Y'	∧ , ε	$,) , ∨
Z	¬ , (, b	$,) , ∨, ∧
T	(, b	$,) , ∨, ∧

(01.5pts).	b	¬	∨	∧	(	)	$
X	*YX'*	*YX'*			*YX'*		
X'			∨ *YX'*			ε	ε
Y	ZY'	ZY'			ZY'		
Y'			ε	∧ *ZY'*		ε	ε
Z	T	¬ T			T		

4. Simulation du fonctionnement de l'analyseur LL(1) sur le mot **(1.5 pt).** : **¬ (b ∨ b) ∧ b**

Chaine	Pile	Action
¬ (b ∨ b) ∧ b$	$X	Dépiler, Empiler X'Y
¬ (b ∨ b) ∧ b$	$X'Y	Dépiler, Empiler Y'Z
¬ (b ∨ b) ∧ b$	$X'Y'Z	Dépiler, Empiler *T* ¬
¬ (b ∨ b) ∧ b$	$X'Y' *T* ¬	Dépiler, Avancer
(b ∨ b) ∧ b$	$X'Y' *T*	Dépiler, Empiler)X(
(b ∨ b) ∧ b$	$X'Y')X(	Dépiler, Avancer
b ∨ b) ∧ b$	$X'Y')X	Dépiler, Empiler X'Y
b ∨ b) ∧ b$	$X'Y')X'Y	Dépiler, Empiler Y'Z
b ∨ b) ∧ b$	$X'Y')X'Y'Z	Dépiler, Empiler *T*
b ∨ b) ∧ b$	$X'Y')X'Y'T	Dépiler, Empiler b
b ∨ b) ∧ b$	$X'Y')X'Y'b	Dépiler, Avancer
∨ b) ∧ b$	$X'Y')X'Y'	Dépiler, Empiler ε
∨ b) ∧ b$	$X'Y')X'	Dépiler, Empiler *X'Y*∨
∨b) ∧ b$	$X'Y')X'Y∨	Dépiler, Avancer
b) ∧ b$	$X'Y')X'Y	Dépiler, Empiler Y'Z
b) ∧ b$	$X'Y')X'Y'Z	Dépiler, Empiler T
b) ∧ b$	$X'Y')X'Y'T	Dépiler, Empiler b
b) ∧ b$	$X'Y')X'Y'b	Dépiler, Avancer
) ∧ b$	$X'Y')X'Y'	Dépiler, Empiler ε
) ∧ b$	$X'Y')X'	Dépiler, Empiler ε
) ∧ b$	$X'Y')	Dépiler, Avancer

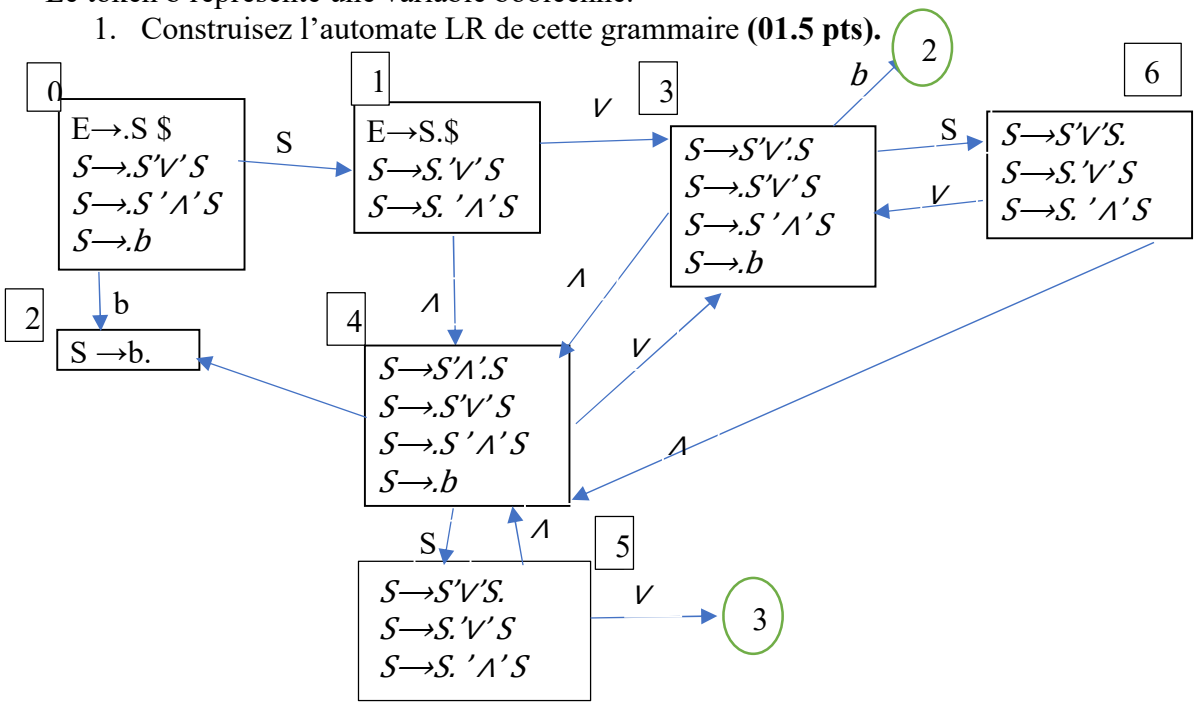
Λ b$	$X'Y'	Dépiler, Empiler ***Y'ZΛ***
Λ b$	$X'Y' ***ZΛ***	Dépiler, Avancer
b$	$X'Y' ***Z***	Dépiler, Empiler T
b$	$X'Y' ***T***	Dépiler, Empiler b
b$	$X'Y' ***b***	Dépiler, Avancer
$	$X'Y'	Dépiler, Empiler ***ε***
$	$X'	Dépiler, Empiler ***ε***
$	$	Accépter

Partie 2: Soit la grammaire G suivante, écrite dans le format reconnu par yacc:

0) E→.S $
1) S⟶.S'V'S
2) S⟶.S 'Λ'S
3) S⟶.b

Le token b représente une variable booléenne.

1. Construisez l'automate LR de cette grammaire **(01.5 pts).**



2. Quelle sorte de conflits apparaissent **(01.5 pts).**

	b	V	Λ	$		S
0	d2					1
1	Acc	Acc/ d3	Acc/d4	Acc		
2	r3	r3	r3	r3		
3	d2	D3	D4			6
4	D2	D3	D4			5
5	R1	R1/d3	R1/d4	R1		
6	R2	R2/d3	R2/d4	R2		

3. Proposez une solution pour résoudre ces conflits. Indiquer les modifications à apporter dans le fichier yacc **(1 pt).**.

```
%token b
%left 'V'
%left 'Λ'
%%
L⟶ S'|n'
S⟶ S'V'S
```

```
    /S'^'S
    /b
    ;
%%
```

Université Mohamed Khider – Biskra
Faculté des Sciences Exactes et des
Sciences de la Nature et de la Vie
Département d'Informatique

Module: **Compilation**
Niveau : **3ème Licence**
Année universitaire : **2017/2018**
Documents non autorisés

Rattrapage

Date : 19/06/2018 **Durée : 1h30m**

Exercice 01 (04 pts) : Soit l'alphabet A = {a,b}, définir les expressions régulières pour les langages:

1. L'ensemble des mots non vides contenant au plus une fois la chaine *ab*.
2. Tous les mots ne contenant pas plus que deux *b* consécutifs.
3. Tous les mots non vides ne contenants pas trois *b* consécutifs.
4. Ensemble des mots contenant la chaine aab.

Exercice 02 (04 pts) :

Dans le langage html, une balise pour une image à la forme suivante :
<IMG SRC="images/image.gif" ALIGN=center ALT="Texte rempla¸cant l'image" TITLE="Mon image" WIDTH=150 HEIGHT=70>
Les mot-clés désignant des attributs (SRC, TITLE, ...) peuvent être écrits en minuscules. Les attributs ne sont pas forcément donnés dans cet ordre.
Ecrire un analyseur lexical, dans le format reconnu par lex, qui prend en entrée un document HTMLet effectue les traitements suivants :

1. Supprime toutes les images du document
2. Compte le nombre d'images au format gif.

Attention : le document peut contenir d'autres balises délimitées par les signes < et >.

Exercice 03 (06 pts) : Soit la grammaire G d'axiome *S* dont les règles de production sont :

$S \rightarrow ABC|DAD$

$A \rightarrow Aa|\varepsilon$

$B \rightarrow Bb|\varepsilon$

$C \rightarrow Ce|\varepsilon$

$D \rightarrow Dd|f$

1. Il est évident que cette G n'est pas LL(1) : pourquoi (en détail)? Transformez-la en Grammaire G1de type LL(1).
2. Construire la table d'analyse LL(1) pour G1.
3. Analyser la chaine « *abedf $*» par un analyseur LL(1)

Exercice 04 (06 pts) : Considérons la grammaire suivante :

```
%%
S : A B
;
B : 'b' A B
| /* epsilon*/
;
A : '(' S ')'
| 'a'
;
%%
```

1. Construire l'automate des LR(0)-items, en indiquant pour chaque état la liste complète de ses items.
2. Calculer pour chaque symbole non terminal de cette grammaire les ensembles Premier et Suivant correspondants.
3. La grammaire est-elle LR(0), SLR(1) ? Justifiez vos réponses.

Université Mohamed Khider – Biskra
Faculté des Sciences Exactes et des
Sciences de la Nature et de la Vie
Département d'Informatique

Module: **Compilation**
Niveau : **3ème Licence**
Année universitaire : **2017/2018**
Documents non autorisés

Corrigé type de rattrapage

Date : 19/06/2018 **Durée : 1h30m**

Exercice 01 (04 pts) : Soit l'alphabet A = {a,b}, définir les expressions régulières pour les langages:

1. L'ensemble des mots non vides contenant au plus une fois la chaine *ab*.
2. Tous les mots ne contenant pas plus que deux *b* consécutifs :a*(b| ε)a(b| ε)a*|b|bb
3. Tous les mots non vides ne contenants pas trois *b* consécutifs: a*(ba*b|a)a(b|a)a*
4. Ensemble des mots contenant la chaine aab : (ab)*

Exercice 02 (04 pts) :

Dans le langage html, une balise pour une image à la forme suivante :
<IMG SRC="images/image.gif" ALIGN=center ALT="Texte rempla¸cant l'image" TITLE="Mon image" WIDTH=150 HEIGHT=70>
Les mot-clés désignant des attributs (SRC, TITLE, ...) peuvent être écrits en minuscules. Les attributs ne sont pas forcément donnés dans cet ordre.
Ecrire un analyseur lexical, dans le format reconnu par lex, qui prend en entrée un document HTMLet effectue les traitements suivants :

1. Supprime toutes les images du document
2. Compte le nombre d'images au format gif.

Attention : le document peut contenir d'autres balises délimitées par les signes < et >.

Exercice 03 (06 pts) : Soit la grammaire G d'axiome *S* dont les règles de production sont :

$S \rightarrow ABC|DAD$
$A \rightarrow Aa|\varepsilon$
$B \rightarrow Bb|\varepsilon$
$C \rightarrow Ce|\varepsilon$
$D \rightarrow Dd|f$

1. Il est évident que cette G n'est pas LL(1) : pourquoi (en détail)?
   Récursivité à gauche dans les règles : **(0.5 pts)**
   $A \rightarrow Aa|\varepsilon$
   $B \rightarrow Bb|\varepsilon$
   $C \rightarrow Ce|\varepsilon$
   $D \rightarrow Dd|f$
2. Transformez-la en Grammaire G1 de type LL(1) : **(01.5 pts)**
   $S \rightarrow ABC|DAD$
   $A \rightarrow aA|\varepsilon$
   $B \rightarrow bB|\varepsilon$
   $C \rightarrow eC|\varepsilon$
   $D \rightarrow fD'$
   $D' \rightarrow dD'|\ \varepsilon$
3. Construire la table d'analyse LL(1) pour G1

(1.5 pts)	Premiers	Suivants
S	a,b,e,f, ε	$
A	a, ε	$,b,e,f
B	b, ε	$,e,
C	e, ε	$

D	F	$,a,f
D’	d, ε	$,a,f

(1.5 pts)	a	b	D	E	F	$
S	ABC	ABC	ABC	ABC	DAD	ABC
A	aA	ε		ε	ε	ε
B		bB		ε		ε
C				eC		ε
D					fD'	
D’	ε		dD'		ε	ε

4. Analyser la chaine « *abedf $*» par un analyseur LL(1) :La chaine n’appartient pas au langage **(1.5 pts)**

Exercice 04 (06 pts) : Considérons la grammaire suivante :

```
%%
S : A B
;
B : 'b' A B
| /* epsilon*/
;
A : '(' S ')'
| 'a'
;
%%
```

1. Construire l’automate des LR(0)-items, en indiquant pour chaque état la liste complète de ses items.
2. Calculer pour chaque symbole non terminal de cette grammaire les ensembles Premier et Suivant correspondants.
3. La grammaire est-elle LR(0), SLR(1) ? Justifiez vos réponses.

Université Mohamed Khider – Biskra
Faculté des Sciences Exactes et des
Sciences de la Nature et de la Vie
Département d'Informatique

Module: **Compilation**
Niveau : **3ème Licence**
Année universitaire : **2017/2018**
Documents non autorisés

Corrigé type du Rattrapage

Fait le : 04/03/2014 **Durée : 1h30m**

Exercice 01 (06 pts) :

1. ([a-zA-Z0-9]|\-_)+@ ([a-zA-Z0-9]|\-_)+
2. (0|1)*0(0|1)*0(0|1)*
3. 1(0|ε)1*(0|ε)1*(0|ε)1*| 01*(0|ε)1*(0|ε)1*
4. a*(b*((aa)a*)*)*(a| ε)

Exercice 03 (04 pts) :

1) Ecrire un programme en lex qui supprime une suite d'espaces et les remplace par un seul espace.

```
( )* printf(" ");
%%
int main(int argc, char *argv[])
{yyin = fopen(argv[1], "r");
yylex();
fclose(yyin);}
```

2) Ecrire un programme en Flex qui compte le nombre de caractères, de mots et de lignes d'un texte %{ int nbm = 0, nbl = 0, nbc = 0;

```
%}
mot     [a-zA-Z]+
%%
{mot}    nbm++; nbc+=yyleng ;
\n        nbl++; nbc++ ;
.         nbc++ ;
%%
main()
{ yylex();
```

printf("Il y a %d voyelles, %d consonnes et %d ponctuations.\n",nbm, nbc, nbl); }

Exercice 02 (10 pts) :

Soit G la grammaire suivante :

A →da|acB
B →abB|daA|Af
G est-elle LL(1) ? justifier votre réponse. Non, il existe une factorisation indirecte dans la règle :
B →abB|daA|Af dont Premiers (A) ∩Premiers (B)≠Ø **0.5pt**
Sinon la transformer pour qu'elle le soit (grammaire G').

A →da|acB
B →abB|daA|(da|acB)f
B →abB|daA|daf|acBf
B →abB|acBf|daA|daf
B →a(bB|cBf)|da(A|f)
B →aB'|daB''
B' →(bB|cBf)
B ''→(A|f)
G' **1.5pts** :
A →da|acB
B →aB'|daB''
B' →bB|cBf

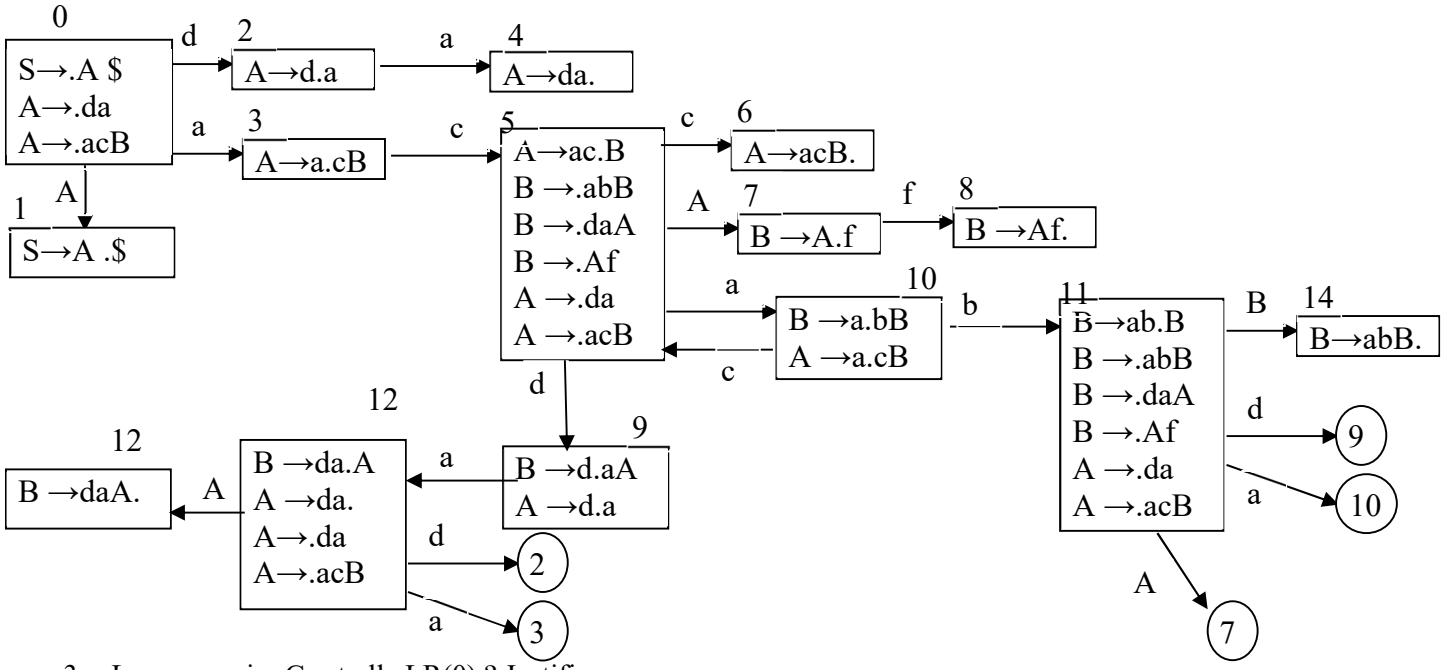
B ‘’→A|f

1. Construire la table d'analyse LL(1).

01pt	Premiers	Suivants
A	d,a	$,f
B	d,a	$,f
B’	b,c	$,f
B’’	d,a,f	$,f

2pts	a	b	c	d	f	$
A	acB			da		
B	aB’			daB’’		
B’		bB	cBf			
B’’	A			A	f	

2. Construire l'automate LR pour G. **(02.5 pts)**



3. La grammaire G est-elle LR(0) ? Justifier.

SLR(1)	a	b	c	d	f	$	**(02.5 pts)**	A	B
0	d3			d2				1	
1						Acc			
2	d4								
3			d5						
4					r1	r1			
5	d10							6	7
6					r2	r2			
7					d8				
8					r5	r5			
9	d12								
10		d11	d5						
11	d10			d9				7	14
12	d3			d2	r1	r1		13	
13					r4	r4			
14					r3	r3			

	Premiers	Suivants
A	a,d	$,f
B	a,d	$,f