Examen « Compilation »
Enseignant : Karim Baïna
Durée = 2H00
(Seuls les documents de cours et les TP sont autorisés !!)
NB : les **réponses *directes*** et ***synthétiques*** seront appréciées

Exercice I : QCM 5 pts (à rendre avec votre copie !!)

Pour chaque concept/question, remplissez la case de la colonne des choix uniques correspondante par un choix qui soit le plus adéquat :

Concept/Question	Choix unique
(**1**) Représentation intermédiaire linéaire	
(**2**) Automate à Piles	
(**3**) Grammaire régulière	
(**4**) Récursivité gauche	
(**5**) Ambiguïté	
(**6**) Erreur de parenthésage non équilibré	
(**7**) LL(1)	
(**8**) LALR	
(**9**) Analyse sémantique	
(**10**) Automate d'état finis	
(**11**) Déterminisme	
(**12**) Représentation intermédiaire graphique	
(**13**) Identificateur erroné	
(**14**) lemme de l'étoile	
(**15**) L2G	

Choix possibles
(**a**) DAG
(**b**) Parseur bottom-up
(**c**) Assembleur
(**d**) Langage régulier
(**e**) Pompage
(**f**) Parseur Top-down
(**g**) Bouclage du parseur LL(1)
(**h**) Langage irrégulier
(**i**) Grammaire linéaire LL(1)
(**j**) Analyse lexicale
(**k**) Grammaire linéaire
(**l**) Code à 2-adresses
(**m**) Analyse syntaxique
(**n**) Erreur de type
(**o**) 2 Arbres syntaxiques

Exercice II : Problème 15 pts

On s'intéresse à l'analyse de logs d'exécution de composants logiciels sur Internet (les Web Services). Pour ce faire on voudrait réaliser un analyseur sémantique qui analyse syntaxiquement les différents logs hétérogènes qui existent et qui en réalise une analyse sémantique en calculant des indicateurs sur la base des logs d'exécution de Web Services.

On définit un événement d'invocation d'une opération de Web Service (**EVENTOP)** comme une structure qui contient : (**1**) le nom de cette opération (**OPERATION : identificateur**), (**2**) l'instant d'invocation de cette opération (**TIME**) qui est sous la forme JJ.MM.AA.hh.mm.ss.mm, (**3**) l'adresse **IP** du client qui a invoqué l'opération, et (**4**) la liste (éventuellement vide) des paramètres input de l'opération invoquée (**PARAM** : chaîne ne contenant pas de caractère #) fournis par le client à cette opération invoquée. La structure d'un événement est la suivante # OPERATION:TIME :IP(:PARAM)*#

Exemple d'événement :
#orderCD:16.03.2005.18.25.30.25:81.192.21.159:SUFIS'S DREAM:ASIA PRODUCTION#

Exercice II-A (Analyse Lexicale) 4pts

(a) Donner les automates déterministes (sous-entendus sans ε-transition) qui reconnaissent les unités lexicales (ou tokens) : **IP**, **OPERATION**, **TIME**, **PARAM**, et **EVENTOP**.

(b) Programmer le scanner qui reconnaît le token **EVENTOP** sous FLEX

Exercice II-B (Analyse Syntaxique) 5pts

Un log de Web Service (**LOG**) est une suite éventuellement vide d'événements. Un événement existe sous deux formes différentes : **EVENTOP**: #OPERATION:TIME:IP(:PARAM)*# (préalablement étudié) et **EVENTIP:** #IP:OPERATION:TIME(:PARAM)*#.
(a) Donner une grammaire $\mathbf{G_{LOG}}$ LALR simple (dont le start est $\mathbf{S_{LOG}}$) qui reconnaît le langage $\mathbf{L(S_{LOG})}$ des logs d'exécution de Web Services, en vous basant sur les résultats de l'Exercice I.
(b) Quel est le type du langage $\mathbf{L(S_{LOG})}$, démontrez-le.
(c) Donner la grammaire LL(1) équivalente à $\mathbf{G_{LOG}}$.
(d) Programmer le parseur LALR (bottom-up) qui reconnaît le langage $\mathbf{L(S_{LOG})}$ sous BISON.

Exercice II-C (Analyse Sémantique) 6pts

Afin d'analyser le logs de notre Web Service, nous désirons calculer les indicateurs **I1**-**I3** suivants :
I1- le nombre de fois qu'une adresse IP donnée invoque notre Web Service en question
I2- le nombre de fois qu'une opération donnée est invoquée chaque jour
I3- la moyenne du nombre de fois qu'une opération donnée est invoquée par jour

(a) Donner une grammaire attribuée et compléter l'analyse lexicale par des actions sémantiques pour calculer en une seule passe les trois indicateurs **I1-I3**.
(b) Programmer l'analyseur sémantique implantant ce système sémantique sous BISON.

Examen « Compilation I »
Enseignant : Karim Baïna
Durée = 1H30

(Seuls les documents de Cours et de TD sont autorisés !!)
NB : les **réponses *directes*** et ***synthétiques*** seront appréciées

Nom :……
Prénom :……..

Exercice I : QCM 5 pts (à rendre avec votre copie !!)

Pour chaque concept/question, remplissez la case de la colonne des choix uniques correspondante par un choix qui soit le plus adéquat :

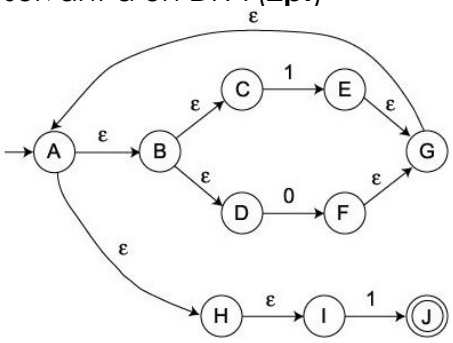
Concept/Question	Choix unique	Choix possibles
(**1**) A= < S, Σ, δ, s_0, F > où S ∩ F ≠ ∅		**(a) Assembleurs**
(**2**) Automate à Piles		(**b**) Analyse syntaxique
(**3**) Grammaire régulière		(**c**) Pompage
(**4**) card(ε-fermeture(s_0)) > 1		(**d**) Analyse lexicale
(**5**) typedef void * Vector ;		(**e**) Langage irrégulier
(**6**) Erreur de parenthésage non équilibré		(**f**) Grammaire linéaire
(**7**) Automate d'état finis		(**g**) Langage régulier
(**8**) Identificateur erroné		(**h**) Lexème
(**9**) Lemme de l'étoile		(**i**) ε ∈ L
(**10**) Token		(**j**) δ(s_0, ε)= s_1, où $s_0 \neq s_1$
(11) L2G	**(a) « Question résolue »**	(**k**) fermeture de Kleene

Exercice II : Expressions régulières 5pts

1. Soit L un langage fini, démontrez que L est régulier. (**1pt**)
2. Est-ce la réciproque est vraie ? Justifiez ! (**1pt**)
3. Est-ce que l'intersection de deux langages réguliers est un langage régulier ? Justifiez ! (**1pt**)
4. Est-ce que le complémentaire d'un langage régulier L ($\Sigma^* \setminus L$) est un langage régulier ? Justifiez ! (**1pt**)
5. Est-ce que le langage $L=\{n \in IN / n \equiv 0 [16]\}$ est régulier ? Justifiez ! (**1pt**)

Exercice III : Programmation d'automates 10pts

1. Si un automate A(L) reconnaît le langage L en n états et p transitions. Quelle est la complexité en temps de la fonction indicatrice $P_L : \Sigma^* \rightarrow \{0,1\}$ où $P_L(w \in L)=1$ et $P_L(w \notin L)=0$. (**1pt**)
2. Quel est l'intérêt pratique de minimiser un automate A(L) (i.e. trouver un automate A'(L) équivalent à A(L) avec n et p minimaux) ? Donner des exemples de systèmes pour lesquels cette technique est incontournable (**1pt**)
3. Transformez le NFA suivant à un DFA (**2pt**)



4. Trouvez l'expression régulière équivalente à l'automate résultant de III.2. (a) intuitivement et (b) en utilisant l'algorithme vu en cours (**2pt**)
5. Donnez la grammaire linéaire équivalente à l'automate résultant de III.3. (**1pt**)

1. Donner deux manières en langage C de programmer l'expression régulière résultant de III.4 (a) l'une à base de l'automate DFA et (b) l'autre à base de la grammaire de V.1. (**3pt**)

Examen « Compilation II »
Enseignant : Karim Baïna
Durée = 2H00

Seuls les documents de Cours et de TD sont autorisés !!
Le barème est donné seulement à titre indicatif !!
*Les **réponses directes** et **synthétiques** seront appréciées*

Nom :……
Prénom :……..

Exercice I : QCM 5 pts (à rendre avec votre copie !!)

Pour chaque concept/question, remplissez la case de la colonne des choix uniques correspondante par un choix qui soit le plus adéquat :

Concept/Question	Choix unique	Choix possibles
(**1**) DAG	(**b**)	**(a)** démontrer qu' « une grammaire est ambiguë » est décidable mais l'inverse est non décidable
(**2**) bytecode J2ME		(**b**) Représentation……..
(**3**) Grammaire attribuée		(**c**) Analyse Bottom-up
(**4**) Grammaire LL		(**d**) Erreur Syntaxique
(5) **A**corn **R**ISC **M**achine-ARM		(**e**) Représentation……..
(**6**) select * from * ;		(**f**) Analyse Top-down
(**7**) bytecode	(**e**)	(**g**) Erreur Sémantique
(**8**) select T1.A1 from T2 ;		(**h**) Classe d'expression régulière de Σ^*
(**9**) Grammaire LALR		(**i**) actions sémantiques
(**10**) Terminal $t \in T$		(**j**) one-address code
(11) semi-décidabilité	**(a) « *RESOLUE* »**	(**k**) three-address code

Exercice II : Analyse Syntaxique / Contextuelle 10 pts

Soit la grammaire LALR G_{SELECT} du langage sous la forme BNF suivante :

<SELECT> ::= **SELECT** *<PROJECT> <FROM>*
<PROJECT> **'*'** | *<COLUMNS>*
<FROM> **FROM** *<TABS> <FROMAUX>*
<COLUMNS> ::= *<COLUMN> <COLUMNAUX>*
<COLUMNAUX> ::= **ε** | **','** *<COLUMNS>*
<COLUMN> ::= **IDF** *<POINTEDCOLUMN>*
<POINTEDCOLUMN> ::= **ε** | **'.' IDF**
<TABS> ::= **IDF** | **IDF** ',' *<TABS>*
<FROMAUX> ::= **ε** | *<WHERE>* | *<ORDERBY>*
<WHERE> ::= **WHERE** *<EXPBOOL>*
<EXPBOOL> ::= **NOT** *<EXPBOOL>*
| *<EXPBOOL>* **AND** *<EXPBOOL>*
| *<EXPBOOL>* **OR** *<EXPBOOL>*
| *<COLUMN> <OP> <COLUMN>*
<OP> ::= **<** | **<=** | **>** | **>=** | **=** | **<>**

1. Démontrer que la grammaire G_{SELECT} est Ambiguë (a) contre-exemple et (b) causes d'ambiguïté (2 pt)
2. Eliminer l'ambiguïté en se basant sur les mêmes conventions que le cours (2 pts)
3. Eliminer la récursivité gauche de la grammaire G_{SELECT} (2 pt)
4. Rendre la grammaire G_{SELECT} LL(1) (2 pts)
5. Donner quatre défauts ou limitations syntaxiques de la grammaire G_{SELECT} et proposer les solutions pour ces trois défauts (2 pts)

Exercice III : Sémantique et Programmation 6 pts[1]

1. Rendre la grammaire G_{SELECT} attribuée LL(1) (2 pt)
2. Programmer la grammaire attribuée LL(1) en C (2 pt)
3. Programmer la grammaire attribuée LALR en bison ? (2 pt)

[1] *(dont 1 pt optionnel)*

Examen « Compilation I »
Enseignant : Karim Baïna
Durée = 1H30
(Seuls les documents de Cours et de TD sont autorisés !!)
NB : le style ***rigoureux*** *et* ***synthétique*** sera apprécié

Nom :……
Prénom :……..

Exercice I : QCM 5 pts (à rendre avec votre copie !!)

Pour chaque concept/question, remplissez la case de la colonne des choix uniques correspondante par un choix qui soit le plus adéquat :

Concept/Question	Choix unique	Choix possibles
(**1**) A= < S, Σ, δ, s_0, F > où $s_0 \notin F$		**(a) Langage binaire**
(**2**) Automate à Piles		(**b**) Analyse syntaxique
(**3**) Système d'équations		(**c**) deux arbres syntaxiques
(**4**) ε-fermeture(s_0) \ { s_0} $\neq \emptyset$		(**d**) Analyse lexicale
(**5**) Problème semi-décidable		(**e**) Langage hors contexte
(**6**) Erreur : if sans endif (en csh)		(**f**) Grammaire linéaire
(**7**) Automate d'état finis		(**g**) Langage régulier
(**8**) Erreur : /* sans */ (en C)		(**h**) Minimiser un automate
(**9**) Grammaire ambiguë		(**i**) $\varepsilon \notin L$
(**10**) Optimisation en mémoire		(**j**) Vérifier l'ambiguïté d'une grammaire
(11) L1G	**(a) « résolu »**	(**k**) $\delta(s_0, \varepsilon) = s_1$, où $s_0 \neq s_1$

Exercice II : Langages réguliers 10pts

1. Démontrer que pour toute expression régulière α et β :
   a) $\alpha(\beta\alpha)^* = (\alpha\beta)^*\alpha$ (**1pt**)
   b) $\alpha^*(\beta\alpha^*)^* = (\alpha^*\beta)^*\alpha^*$ (**1pt**)
2. Montrez que pour tout langage régulier L_1 et L_2, le langage $L_1 \cap L_2$ est régulier (fermeture des langages réguliers par intersection) (**1pt**)
3. Donner l'expression régulière décrivant le langage L={n∈IN / n ≡ 0 [32]} (**1pt**)
4. Si la complexité en temps de la fonction de transition δ est en $O(1)$ quelle est la complexité en temps de l'algorithme non optimisé de transformation d'un automate non déterministe sans epsilon transition $\mathbf{A_N = <S_N, \Sigma_N, \delta_N, s_{N0}, F_N >}$ en un automate déterministe ? (**1pt**)
5. Soit $(A_i)_{i=0..n}$ une suite d'automates d'états finis sous la forme de la figure 1 (**a**) donner et démontrer la forme régulière générale des langages $L(A_i)_{i=0..n}$ (**b**) quelle est la propriété conservée par la suite $(L(A_i))_{i=0..n}$ (**c**) démontrer cette propriété. (**3pts**)

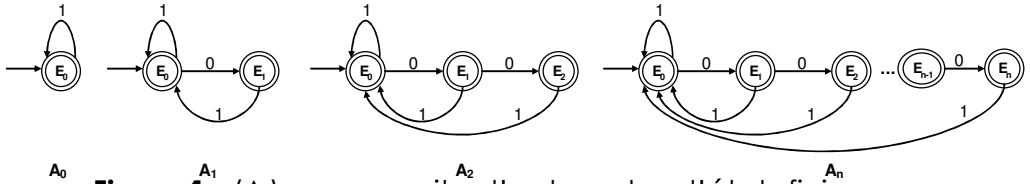


Figure 1 : $(A_i)_{i=1..n}$ une suite d'automates d'états finis

6. Rendre le NFA $A_N=<S_N=\{A..T\},\Sigma_N=\{0,1\},\delta_N,s_{N0}=A,F_N=\{T\}>$ déterministe (**2pts**)

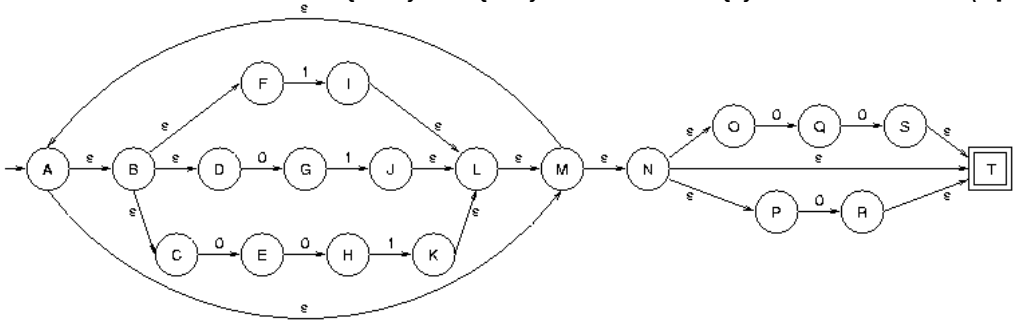


Figure 2 : Automate non déterministe à epsilon transition

Exercice III : Langages hors contextes 5pts

1. Donner et décrire les grammaires hors contexte des langages suivants :
   a. L1 = { $a^i b^j c^k$ / i≠j ou j≠k} (**1pt**)
   b. L2 = { $a^i b^j c^k$ / j=i+k} (**1pt**)
   c. L3 = {w ∈ {a,b}* / w = vv^{-1} ou w = v$\bar{v}^{-1}$, où v ∈ {a,b}*}[1] (**1pt**)
2. Soit la grammaire suivante G = <T={*a*,*b*,*c*}, NT={A,B,C,D}, S, P={r_1..r_{10}}>

S	::=	**<A> <B>**	(r_1)	| **<C> <D>**	(r_2)
A	::=	*a* **<A>** *b*	(r_3)	| *ab*	(r_4)
B	::=	*c*	(r_5)	| *c* **<B>**	(r_6)
C	::=	*a*	(r_7)	| *a* **<C>**	(r_8)
D	::=	*b* **<D>** *c*	(r_9)	| *bc*	(r_{10})

   a. Quel est le langage L(G), justifier (**1pt**)
   b. Que dire de la grammaire G, justifier, donner des idées de solutions éventuelles (**1pt**)

[1] $\bar{v}$ *dénote le complément de v dans {a,b}* et v^{-1} dénote l'inverse de v dans {a,b}* (ex :* $\overline{abb}$ *=baa, (abb)$^{-1}$=bba,* $\overline{abb}^{-1}$ *=aab)*

Examen « Compilation II »
Enseignant : Karim Baïna
Durée = 2H00

Seuls les documents de Cours et de TD sont autorisés !!
Le barème est donné seulement à titre indicatif !!
*Les **réponses directes** et **synthétiques** seront appréciées*

Nom :……
Prénom :……..

Exercice I : QCM 5 pts (à rendre avec votre copie !!)

Pour chaque concept/question, remplissez la case de la colonne des choix uniques correspondante par un choix qui soit le plus adéquat :

Concept/Question	Choix unique	Choix possibles
(**1**) Control Flow Graph	(**b**)	**(a)** démontrer qu' « une grammaire est ambiguë » est décidable mais l'inverse est non décidable
(**2**) bytecode J2EE		(**b**) Représentation……..
(**3**) Grammaire attribuée		(**c**) Analyseur Ascendant
(**4**) Grammaire LL		(**d**) Erreur Syntaxique
(**5**) **A**corn **R**ISC **M**achine-ARM		(**e**) Représentation……..
(**6**) select * from * ;		(**f**) Analyseur Descendant
(**7**) bytecode	(**e**)	(**g**) Erreur Sémantique
(**8**) select T1.A1 from T2 ;		(**h**) Erreur Lexicale
(**9**) Grammaire LR		(**i**) actions sémantiques
(**10**) Commentaire C non fermé (/* sans */)		(**j**) one-address code
(11) semi-décidabilité	**(a) « *RESOLUE* »**	(**k**) three-address code

Exercice II : Analyse Syntaxique / Contextuelle 10 pts

Soit la grammaire LALR G_{pcsh} du langage sous la forme BNF suivante :

<SCRIPTPCSH> ::= **<HEAD>** **<INSTLIST>**
<HEAD> ::= "#!/bin/pcsh" *RC*
<INSTLIST> ::= **<INST>** **<INSTLISTAUX>**
<INSTLISTAUX> ::= **ε** | *RC* **<INSTLIST>**
<INST> ::= **<ECHO>** | **<ASSIGN>** | **<IF>** | **<FOREACH>** | **<WHILE>** | *COMMENT*
<ECHO> ::= echo **<ECHOAUX>**
<ECHOAUX> ::= '$'IDF | **<ELT>**
<ASSIGN> ::= '@' *IDF* '=' **<EXPNUM>** | "set" *IDF* = *STRING*
<IF> ::= "if" '(' **<EXPBOOL>** ')' then *RC* **<INSTLIST>** **<ELSE>** *RC* "endif"
<ELSE> ::= **ε** | *RC* "else" *RC* **<INSTLIST>**
<FOREACH> ::= "foreach" *IDF* '(' **<ELTLIST>** ')' *RC* **<INSTLIST>** *RC* "end"
<WHILE> ::= "while" '(' **<EXPBOOL>** ')' *RC* **<INSTLIST>** *RC* "end"
<EXPNUM> ::= *NUM* | ' $'*IDF* | **<EXPNUM>** *OPNUM NUM* | **<EXPNUM>** *OPNUM '$'IDF*
<EXPBOOL> ::= ' $'IDF | **<EXPBOOL>** *OPBOOL* **<EXPBOOL>** | ! '(**<EXPBOOL>** ')' | ' $'*IDF COMP* **<ELT>**
<ELTLIST> ::= **<ELT>** **<ELTLISTAUX>**
<ELTLISTAUX> ::= **ε** | **<ELTLIST>**
<ELT> ::= *STRING* | *NUM*

Où respectivement la description des terminaux est la suivante : RC (retour chariot), COMMENT (commentaire c-shell sur une ligne : toute suite de caractères commençant par une #) IDF (identificateur), OPNUM (opérateur arithmétique : *, +, -, /, %), STRING (chaîne de caractères entre apostrophes ' '), OPBOOL (opérateur logique : &&, ||, !), IDFORNUM (IDF ou NUM), COMP (opérateur relationnel : ==, <=, >=, !=). On utilisera la sémantique usuelle des instructions C-SHELL.

1. Démontrer que la grammaire G_{pcsh} est Ambiguë (a) contre-exemple et (b) causes d'ambiguïté (2 pt)
2. Eliminer l'ambiguïté en se basant sur les mêmes conventions que le cours (2 pts)
3. Eliminer la récursivité gauche de la grammaire G_{pcsh} (2 pt)
4. Rendre la grammaire G_{pcsh} LL(1) (2 pts)
5. Donner trois défauts ou limitations syntaxiques de la grammaire G_{pcsh} et proposer les solutions pour ces trois défauts (2 pts)

Exercice III : Sémantique et Programmation 6 pts[1]

1. Rendre la grammaire G_{pcsh} attribuée LL(1) (2 pt)
2. Programmer la grammaire attribuée LL(1) en C (2 pt)
3. Programmer la grammaire attribuée LALR en bison ? (2 pt)

[1] *(dont 1 pt optionnel)*

Examen « Compilation I »
Enseignant : Prof. Karim BAÏNA
Durée = 1H30

(Seuls les documents de Cours et de TD sont autorisés !!)
NB : le style ***rigoureux*** *et* ***synthétique*** sera apprécié
Les réponses sont à rendre sur cette même copie

Nom :……
Prénom :…….

Exercice I : Questions de cours (5 pts)

(réponse fausse = 0)

Pour chaque concept/question, remplissez la case de la colonne des choix uniques correspondante par un choix qui soit le plus adéquat

Concept/Question	Choix unique	Choix possibles
(**1**) DFA= < S, Σ, δ, s_0, F > où $s_0 \notin F$		**(a) Langage d'assemblage**
(**2**) pompage		(**b**) peudo-équivalence (asymétrie)
(**3**) analyse lexicale		(**c**) Σ^* où Σ = {a, b}
(**4**) ε-fermeture(s_0) \ { s_0} ≠ ∅		(**d**) scanning
(**5**) analyse syntaxique		(**e**) (ab)* b
(**6**) ε-fermeture		(**f**) parsing
(**7**) a (ba)*		(**g**) irrégularité d'un langage
(**8**) résiduels à gauche		(**h**) minimiser un automate de contrôle
(**9**) (a+b)*		(**i**) ε ∉ L
(**10**) mémoire portable optimisée		(**j**) calcul des suffixes de suffixes
(11) L2G	**(a) « résolu »**	(**k**) δ(s_0, ε)= s_1, où $s_0 \neq s_1$

Exercice II : Vérifier et Justifier (5 pts)

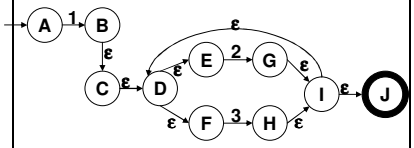
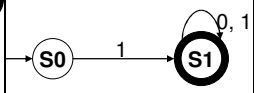
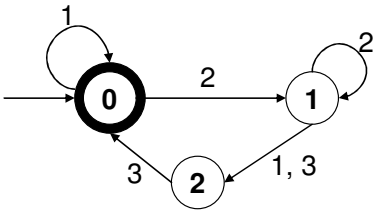
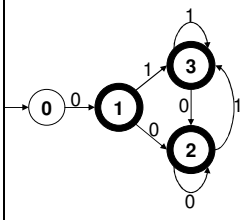
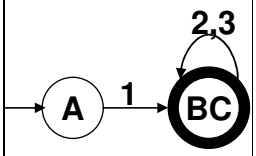
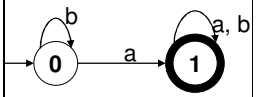
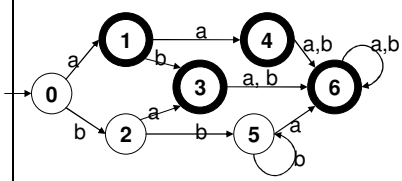
(réponse fausse=-0.5)

1) (a une lettre ⇒ le langage {a*} est régulier) ? Vrai ☐ Faux ☐
……..
2) (Le langage L est fini ⇒ L est régulier) ? Vrai ☐ Faux ☐
……..
3) (L est régulier ⇒ le langage $\Sigma^* \setminus L$ est régulier)? Vrai ☐ Faux ☐
……..
……..
4) (L1 et L2 sont réguliers ⇒ le langage L1 ∩ L2 est régulier) ? Vrai ☐ Faux ☐
……..
……..
……..
5) (le langage L1 ∪ L2 est régulier ⇒ L1 et L2 sont réguliers) ? Vrai ☐ Faux ☐
……..
……..
……..
6) (L* est régulier ⇒ le langage L est régulier) ? Vrai ☐ Faux ☐
……..
……..
……..

Exercice III : Transformations régulières (10 pts)

(réponse fausse=-1)

(Q1) Déterminer les relations d'équivalence 1-1 entre un objet source (de la colonne 1) et un objet cible (de la colonne 3). **(Q2)** Désigner un seul algorithme de construction de cette équivalence (**A_1 :** déterminisation, **A_2 :** Glushkov, **A_3 :** recherche de chemin dans un automate **A_4 :** équations linéaires, **A_5 :** minimisation par séparation des états, **A_6 :** minimisation par calcul des résiduels à gauche). **(Q3)** Démontrer cette équivalence en appliquant l'algorithme choisi en **(Q2)**.

colonne 1	(Q1)	colonne 3	(Q2)	(Q3)
Objet Source	**Objet cible choisi (A..E)**	**Objet Cible**	**Algo. choisi (A_1..A_6)**	**Démonstration**
		**(A)** 		
		**(B)** 		
$0^+ (0 + 1)^*$		**(C)** 		
$1^*1(0+1)^*$		**(D)** 		
		**(E)** $(1 \mid 2^+(1 \mid 3)\ 3)^*$		

Examen « Compilation II »
Enseignant : Karim Baïna
Durée = 2H00

Seuls les documents de Cours et de TD sont autorisés !!
Le barème est donné seulement à titre indicatif !!
*Les **réponses directes** et **synthétiques** seront appréciées*

Nom :……
Prénom :……..

Exercice I : QCM 5 pts (à rendre avec votre copie !!)

Pour chaque concept/question, remplissez la case de la colonne des choix uniques correspondante par un choix qui soit le plus adéquat :

Concept/Question	Choix unique	Choix possibles
(**1**) Bytecode Java	(**b**)	**(A)** démontrer qu' « une grammaire est ambiguë » est décidable mais l'inverse est non décidable
(**2**) **ADDOP** REG1, REG2		(**B**) Représentation……..
(**3**) Nombre de registres nécessaire pour un expression arithmétique		(**C**) Analyseur Ascendant
(**4**) Grammaire LL		(**D**) Erreur Syntaxique
(**5**) **A**corn **R**ISC **M**achine-ARM		(**E**) Représentation……..
(**6**) select * from * ;		(**F**) Analyseur Descendant
(**7**) pseudo-code	(**E**)	(**G**) Erreur Sémantique
(**8**) select T1.A1 from T2 ;		(**H**) Erreur Lexicale
(**9**) Grammaire LR		(**I**) Attribut nécessaire à la génération de pseudo-code
(**10**) Commentaire C non fermé (/* sans */)		(**J**) two-address code
(11) semi-décidabilité	**(A)** « ***résolue*** »	(**K**) three-address code

Exercice II : Analyse Syntaxique / Contextuelle 10 pts

Soit la grammaire LALR $G_{pseudoc}$ du langage décrite sous la forme BNF ci-après.

1. Démontrer que la grammaire $G_{pseudoc}$ est Ambiguë (a) contre-exemple et (b) causes d'ambiguïté (2 pt)
2. Eliminer l'ambiguïté en se basant sur les mêmes conventions que le cours (2 pts)
3. Eliminer la récursivité gauche de la grammaire $G_{pseudoc}$ (2 pt)
4. Rendre la grammaire $G_{pseudoc}$ LL(1) (2 pts)
5. Donner trois défauts ou limitations syntaxiques de la grammaire $G_{pseudoc}$ et proposer les solutions pour ces trois défauts (2 pts)

```
<programme>              ⟶  <liste de déclarations> <liste de fonctions>
<liste de déclarations>  ⟶  ε | <déclaration> ; <liste de déclarations>
<déclaration>            ⟶  int identificateur | static int identificateur
                             | int identificateur[nb_entier] | static int identificateur[nb_entier]
                             | <prototype>
<prototype>              ⟶  <type> identificateur( <suite de types> )
<type>                   ⟶  void | int
<suite de types>         ⟶  void | <liste de types>
<liste de types>         ⟶  int | int, <liste de types>
<liste de fonctions>     ⟶  <fonction> | <fonction> <liste de fonctions>
<fonction>               ⟶  <en-tête> { <corps> }
<en-tête>                ⟶  <type> identificateur( <suite de paramètres> )
<suite de paramètres>    ⟶  void | <liste de paramètres>
<liste de paramètres>    ⟶  <paramètre> | <paramètre> , <liste de paramètres>

<paramètre>              ⟶  int identificateur
                             | int identificateur[nb_entier]
<corps>                  ⟶  <liste de déclarations> <liste d'instructions>
<liste d'instructions>   ⟶  ε | <instruction> ; <liste d'instructions>
<instruction>            ⟶  identificateur = <expression>
                             | identificateur[ <expression> ] = <expression>
                             | exit(nb_entier) | return <expression> | return
                             | if( <expression> ) <instruction> else <instruction>
                             | if( <expression> ) <instruction>
                             | while( <expression> ) <instruction>
                             | write_string(chaîne) | write_int( <liste d'arguments> )
                             | read_int identificateur
                             | { <liste d'instructions> }
                             | <expression>
<expression>             ⟶  <expression> <opérateur binaire> <expression>
                             | - <expression> | ! <expression> | ( <expression> )
                             | identificateur | nb_entier
                             | identificateur( <suite d'arguments> )
                             | identificateur[ <expression> ]
<suite d'arguments>      ⟶  ε | <liste d'arguments>
<liste d'arguments>      ⟶  <expression> | <expression> , <liste d'arguments>
<opérateur binaire>      ⟶  > | < | == | <= | >= | != | + | - | * | / | % | || | &&
```

Les commentaires sont compris entre /* et */ et ne s'imbriquent pas. Les blancs sont interdits au milieu d'un mot clé, ignorés ailleurs. Un identificateur est une suite de lettres, et une constante entière nb_entier est une suite de chiffres. Le lexème virgule, joue le rôle de séparateur d'identificateurs. Le point-virgule ; joue le rôle de terminateur d'instruction. Les opérateurs relationnels sont ==,=, <=, >=, <, >. Les opérateurs logiques sont || (ou), && (et), et ! (non). L'affectation est notée =. Les opérateurs arithmétiques sont +, -, *, % et /. Le moins unaire est aussi noté -. Parenthèses (), crochets [], accolades {} sont des lexèmes utilisés comme en C. Pour comprendre d'autres éléments de cette grammaire, on utilisera la sémantique usuelle des instructions du langage C.

Exercice III : Sémantique et Programmation 6 pts[1]

1. Rendre la grammaire $G_{pseudoc}$ attribuée LL(1) (2 pt)
2. Programmer la grammaire attribuée LL(1) en C (2 pt)
3. Programmer la grammaire attribuée LALR en bison ? (2 pt)

[1] *(dont 1 pt optionnel)*