

Université Paris 7 — IUP 1^{ère} année
Examen de compilation
29 mai 2002 — Durée : 2h00

*La notation tiendra compte de la clarté des justifications fournies.
Documents de cours, TD, TP autorisés.*

Exercice 1 (5pts) Répondre par vrai ou faux aux questions suivantes. Barème : 0,5 pour une réponse correcte, -0,5 pour une réponse incorrecte et 0 en cas d'absence de réponse. Il n'est pas demandé de justification.

	Vrai	Faux
1. Une grammaire ambiguë ne peut être ni LL(1), ni SLR(1).....	<input type="checkbox"/>	<input type="checkbox"/>
2. Le logiciel yacc permet l'utilisation de grammaires ambiguës.	<input type="checkbox"/>	<input type="checkbox"/>
3. On peut décrire par une expression régulière <code>lex</code> tout langage engendré par une grammaire LL(1).	<input type="checkbox"/>	<input type="checkbox"/>
4. Toute grammaire LR(0) est LL(1).	<input type="checkbox"/>	<input type="checkbox"/>
5. Une grammaire récursive droite ne peut pas être SLR(1).....	<input type="checkbox"/>	<input type="checkbox"/>
6. Le premier état de l'automate LR(0) d'une grammaire G contient toutes les règles de G , le marqueur étant au début de chaque règle.....	<input type="checkbox"/>	<input type="checkbox"/>
7. Une grammaire non factorisée ne peut pas être LR(0).....	<input type="checkbox"/>	<input type="checkbox"/>
8. Il est possible de calculer tous les attributs d'une grammaire S-attribuée lors d'un parcours en profondeur de l'arbre de dérivation.	<input type="checkbox"/>	<input type="checkbox"/>
9. L'ensemble des mots bien parenthésés sur l'alphabet $\{(,)\}$ peut être décrit par une expression <code>lex</code> utilisant seulement les opérateurs concaténation, $*$, et $ $	<input type="checkbox"/>	<input type="checkbox"/>
10. Tout langage construit à partir des caractères en appliquant un nombre fini de fois la concaténation et les opérateurs $*$, $ $ est engendré par une grammaire.	<input type="checkbox"/>	<input type="checkbox"/>

Exercice 2 (5pts) Écrire un source `lex` permettant d'afficher dans un programme C toutes les lignes contenant un commentaire complet, et uniquement celles-ci. On rappelle qu'un commentaire C commence par `/*`, se termine par `*/`, et est hors de toute chaîne de caractères constante, délimitée par les caractères `"`.

Attention, si une ligne contient un commentaire complet, le programme devra afficher la ligne tout entière (et pas seulement le commentaire).

Exercice 3 (10 pts) Ici, toute affirmation insuffisamment justifiée sera considérée comme fausse. Soit G la grammaire suivante :

- $S \longrightarrow TN$ (1)
- $T \longrightarrow \text{char}$ (2)
- $T \longrightarrow \text{int}$ (3)
- $T \longrightarrow \text{double}$ (4)
- $N \longrightarrow N[\text{nb}]$ (5)
- $N \longrightarrow \text{id}$ (6)

Les terminaux sont `char`, `int`, `double`, `nb`, `id`, [et] .

1. Donner l'arbre de dérivation de `int id [nb] [nb]`.
2. La grammaire est-elle ambiguë?
3. La grammaire est-elle LL(1)?
4. Construire l'automate LR(0) de la grammaire.
5. La grammaire est-elle LR(0)? SLR(1)?
6. Simuler le comportement d'un analyseur syntaxique montant sur l'entrée `int id [nb] [nb]`. On précisera la suite de ses actions, l'évolution de la pile d'états et celle de l'entrée.
7. À partir de cette question et jusqu'à la fin de l'exercice, on considère la grammaire G augmentée des deux règles suivantes :

$$N \longrightarrow *N \quad (7)$$

$$N \longrightarrow (N) \quad (8)$$

où `*`, `)`, et `(` sont trois nouveaux terminaux. Montrer que la grammaire ainsi obtenue n'est pas SLR(1). Il n'est pas demandé de reconstruire l'automate complet, mais seulement d'exhiber un conflit. Donner un exemple d'entrée qui met en évidence ce conflit SLR(1).

8. Sur l'entrée trouvée à la question précédente, expliquer quel aurait dû être le choix correct d'un analyseur syntaxique ascendant.
9. On interprète les mots du langage engendré par la grammaire comme des déclarations du langage C. Les types sont construits à partir des types de base `char`, `int` et `double` en utilisant les crochets pour construire des tableaux, l'étoile pour construire des pointeurs et les parenthèses pour permettre de changer l'ordre d'évaluation de ces constructeurs.
De quelle façon faut-il lever le conflit trouvé à la question précédente de manière à ce que les priorités des opérateurs `*` (pointeur) et `[]` (tableau) soient correctement interprétées?
10. Écrire un source `lex` et un source `yacc` pour analyser et décrire en français une déclaration C construite à partir des types `char`, `int` et `double` en utilisant les parenthèses et les constructeurs tableaux et pointeurs. Par exemple, sur l'entrée

`char (*x) [25]`

le programme devra afficher

`x: pointeur vers tableau de 25 caractères`