DÉPARTEMENT D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE

SIGLE DU COURS: IFT 6820 (H2006)

NOM DU PROFESSEUR: Philippe Langlais

TITRE DU COURS: Langages de programmation et Compilation

EXAMEN INTRA

Date : lundi 20 février 2006
Heure : 09:30 - 11:30
Salle : P-318 Pavillon principal

DIRECTIVES PÉDAGOGIQUES:

- Inscrivez votre nom et votre code permanent sur le carnet de réponses.
- Documentation autorisée.
- Les réponses sont à rédiger à même le carnet en indiquant clairement à quelle question vous répondez.
- Le barème est donné à titre indicatif et peut-être modifié lors de la correction. Même si 26 points sont à prendre, votre note à cet examen sera d'au plus 25 points.
- La plupart des questions qui vous sont posées appellent à des réponses courtes et précises. La clarté de vos réponses est donc un critère important qui sera considéré lors de la notation. L'élégance d'une solution sera également un critère de notation.

Question 1	/ 7
Question 2	/ 7
Question 3	/ 12
Total	/ 26

Question 1 - Langage et grammaire (7 points)

1-A Écrire une grammaire qui reconnaît les seules chaînes formées sur l'alphabet $\{a, b\}$ de la forme $a^n b^m$ où $0 < n < m$. *abb* et *aabbbbb* sont des exemples de chaînes appartenant au langage décrit, à contrario de *ab* ou *bbbaa*.

1-B Écrire une grammaire qui reconnaît les seules chaînes formées sur l'alphabet $\{a, b\}$ de la forme $a^n b^m$ où $m = 2 \times n, m > 0$. *abb* et *aabbbb* sont des exemples de chaînes appartenant au langage décrit, à contrario de *ab* ou *aabbb*.

1-C Soit la grammaire $G =< \{A\}, \{a\}, \{A \rightarrow AA|aa|aaa\}, A >$.

1-C-1 Quel est le type de cette grammaire ?

1-C-2 Donner une dérivation **droite** de la chaîne *aaaaa*.

1-C-3 Cette grammaire est-elle ambigüe ? Justifiez.

1-C-4 Quel est le langage décrit par G ? Justifiez.

1-C-5 Selon vous, est-ce un langage de type 3 ? Justifiez.

Question 2 - Analyseur syntaxique (7 points)

Écrire un programme complet C ou C++ ou Java capable de reconnaître les chaînes décrites par la grammaire G suivante:

Terme → Atome	Terme → Var	Terme → Entier
Terme → Atome **(** ListeT **)**	ListeT → Terme	ListeT → Terme **,** ListeT

où `Entier`, `Var` et `Atome` désignent respectivement une constante littérale entière, un identificateur commençant par une majucule, et un identificateur commençant par une minuscule.
Faite l'hypothèse d'un analyseur lexical sous la forme d'une fonction `next()` qui a chaque appel instancie les variables globales `lexeme` et `token` avec respectivement le dernier lexème lu et la classe associée.

2-A Sans écrire un analyseur lexical, indiquez `toutes` les catégories de lexèmes dont vous allez avoir besoin pour écrire un analyseur syntaxique de G.

2-B G est-elle une grammaire LL1 ? Dans la négative, indiquez ce qui posera problème à une technique d'analyse descendante prédictive comme celle vue en cours.

2-C Écrire votre analyseur syntaxique (l'analyseur lexical est supposé fourni).

Question 3 - Prolog (12 points)

Sauf indication contraire, dans cet exercice, vous ne présupposerez l'existence d'aucun foncteur. Vous devez donc définir tous ceux que vous souhaitez utiliser. Là où un foncteur vous est demandé, vous pouvez vous aider d'autres foncteurs.

3-A Écrire un foncteur `diff(L1,L2)` qui s'efface si `L2` est la liste dont le i^{eme} élément est égal à la différence entre le i^{eme} élément de `L1` et le $i+1^{eme}$ élément de `L1`.

Par exemple, `([1,3,4,7,2],[-2,-1,-3,5])` s'efface.

3-B Écrire un foncteur `estTrie(L)` qui s'efface si `L` est triée.

Par exemple `estTrie([1,2,6])` et `estTrie([5,2,1])` sont deux buts qui s'effacent.

3-C Dans les questions qui suivent, faites l'hypothèse qu'en cas d'effacements multiples, l'utilisateur demandera toutes les démonstrations (tapera `;` après chaque preuve). Le prédicat `write`, comme son nom l'indique, s'efface en affichant la chaîne qui lui est donnée. Les numéros de règles ne font pas partie du programme.

```
1-  edge(a,b).
2-  edge(a,c).
3-  edge(c,d).
4-  edge(d,e).
5-  edge(d,f).
6-  edge(e,c).
7-  edge(e,g) :- write('6820').
8-  edge(b,h) :- write('ift').
9-  path(X,X,[X]).
10- path(X,Y,[X|L]) :-  edge(X,N),  path(N,Y,L).
```

3-C-1 Quel est l'affichage `exact` produit par Prolog à l'effacement de `path(a,d,P).` ?

3-C-2 Quel est l'affichage `exact` produit par Prolog à l'effacement de `path(c,g,P).` ?

3-C-3 On inverse maintenant l'ordre des règles 6 et 7. Quel est l'affichage `exact` produit par Prolog à l'effacement de `path(c,g,P).` ?

3-D Considérez les expressions arithmétiques décrites par la grammaire:

$$\text{E} \rightarrow \text{plus(E,E)} \quad \text{E} \rightarrow \text{fois(E,E)}$$
$$\text{E} \rightarrow 1|2|...|9 \quad \text{E} \rightarrow a|b|...|z$$

En voici quelques exemples:

```
fois(1,2)
fois(1,plus(2,4))
plus(b,fois(plus(a,3),plus(2,fois(2,c))))
```

Vous devez ici écrire un foncteur `calcule(E,L,R)` qui s'efface si `R` est le résultat du calcul de E sachant une liste associative `L` associant à des atomes une valeur. Voici des exemples d'appels de ce foncteur qui vous précisent la manière dont est spécifiée la liste associant les atomes d'une expression à leur valeur:

```
calcule(fois(2,3),_,6).
calcule(plus(a,b),[[a,1],[b,2]],3).
calcule(plus(a,fois(b,3)),[[a,1],[b,2]],7).
```

Écrivez `calcule` en faisant l'hypothèse que la liste associative contient tous les atomes présents dans l'expression à calculer. Faites au besoin l'hypothèse que les foncteurs `entier(X)` et `atome(X)` s'effacent respectivement lorsque `X` est un entier ou un atome.