CMP1 – Construction des compilateurs
TYLA – Typologie des Langages

EPITA – Promo 2008 – **Aucun document autorisé**

Avril 2006 (1h30)

Une copie synthétique, bien orthographiée, avec un affichage clair des résultats, sera toujours mieux notée qu'une autre demandant une quelconque forme d'effort de la part du correcteur. Une argumentation informelle mais convaincante, sera souvent suffisante.

1 Incontournables

Il n'est pas admissible d'échouer sur une des questions suivantes : **chacune induit une pénalité sur la note finale**.

1. Quelle est le type (de Chomsky) du langage engendré par $S \to aX \quad S \to b \quad X \to Sc$
2. Qu'est-ce que Lex ?
3. Qu'est-ce que Yacc ?

2 Typologie des Langages

1. Apparier chaque auteur avec son langage :

   a. Alan Kay
   b. Andrew Appel
   c. Bjarne Stroustrup
   d. Denis Ritchie
   e. Jean Ichbiah
   f. John Backus
   g. John McCarthy
   h. Kristen Nygaard
   i. Niklaus Wirth
   j. Ole-Johan Dahl

   1. Ada 83
   2. C
   3. C++
   4. FORTRAN
   5. Lisp
   6. Pascal
   7. Simula
   8. Smalltalk
   9. Tiger

2. Quel langage/compilateur a montré au monde les bénéfices des langages de haut-niveau compilés ?
3. À quel langage doit-on `if then else` ?
4. À quel langage doit-on l'orienté objet ?
5. Que sont les multiméthodes ?
6. En C++, comment simule-t-on les multiméthodes ?
7. En Eiffel, lorsqu'une méthode est redéfinie, qu'advient-il de ses préconditions ?

(a) elles ne sont pas modifiables
(b) elles peuvent être affaiblies
(c) elles peuvent être renforcées
(d) elles sont librement modifiables

8. En Eiffel, lorsqu'une méthode est redéfinie, qu'advient-il de ses postconditions ?

(a) elles ne sont pas modifiables
(b) elles peuvent être affaiblies
(c) elles peuvent être renforcées
(d) elles sont librement modifiables

3 Construction des Compilateurs

1. Quel est l'avantage de l'algorithme GLR sur LALR ?
2. Pourquoi est-ce aussi un désavantage dangereux ?
3. Que signifie AST en anglais et en français ?
4. Donner la syntaxe abstraite (soit sous la forme d'une grammaire, soit sous la forme d'une hiérarchie orientée objet typée) de la grammaire suivante.

```
<term> ::= <term> ( <term> )       -- Application
         | λ <var> . <term>        -- Abstraction
         | <var>                   -- Variable
<var>  ::= <identifier>
```

   On utilise `--` pour introduire des commentaires qui bien sûr n'appartiennent pas à la grammaire.
5. Qu'est-ce qu'une table de symboles ?
6. Pourquoi un langage comme le C inclut les "prédéclarations", et pas un autre comme Java ?
7. Étant donné que le programme Tiger suivant est valide, expliquer quelle politique de gestion mémoire le compilateur doit utiliser pour les types.

```
let
  var box :=
    let
      type box = { val: string }
    in
      box { val = "42\n" }
    end
in
  print (box.val)
end
```

8. Soient deux fractions $\frac{a}{b}$, $\frac{c}{d}$, à quelle surprise s'expose-t-on si l'on programme $\frac{a}{b} \leq \frac{c}{d}$ comme `(a/b) <= (c/d)`.
9. À quoi est dû ce comportement, et comment l'éviter ?