TELECOM Nancy

TELECOM Nancy 2ème année
xx yy janvier 2015
Durée : 1h00 mn (h00 - h00)

Examen de consolidation du module TRAD1

Les documents ne sont pas autorisés.
La note finale tiendra compte du soin apporté à la rédaction de la copie :
une copie illisible n'est pas à votre avantage.
Toute réponse écrite au crayon de papier sera effacée par mes soins.
Les réponses non justifiées n'apporteront aucun point à votre note.

EXERCICE 1 : Analyse syntaxique ascendante. (xx points, yy mn)

Soit la grammaire suivante dans laquelle l'alphabet terminal est constitué des symboles de l'ensemble $\{a\ b\ c\}$, et S' étant l'axiome augmenté.

S'	$\longrightarrow$	S	r_0
S	$\longrightarrow$	A B c	r_1
A	$\longrightarrow$	A a	r_2
	$\longrightarrow$	$\wedge$	r_3
B	$\longrightarrow$	A b	r_4

1. Construisez l'automate et la table d'analyse pour un analyseur ascendant LR(1).
2. Cette grammaire est-elle LR(1) ? Est-elle LALR(1) ?

EXERCICE 2 : Arbre abstrait. (4 points, 20 mn)

Soit le programme Java suivant :

```
public class MyString{

 // attributs
 private String content;
 private int nb;

 // constructeur
 public MyString(String content,int i){
    this.content=content;
    nb=i;
 }

 // fonctions
 public String toString(){
    String res="content"+"("+nb+"):="+content+";";
    return res;
 }

 // main
 public static void main(String[] args){
    MyString test=new MyString("test",1);
    System.out.println(test);
 }
}
```

On rappelle qu'un *arbre abstrait* (ou encore AST) est un arbre qui ne garde plus trace des détails de l'analyse syntaxique, c'est à dire qu'il est indépendant de la grammaire, mais il mémorise la structure du programme.

On vous demande de dessiner un AST de ce programme.
Pour cela, la racine pourra être le nom de la classe (MyString), et vous pourrez créer les noeuds ATTR pour les attributs, CONS pour les constructeurs, FUNCTS pour les fonctions et MAIN pour la fonction main (d'autres noeuds seront certainement à définir...) Vous dessinerez ces 4 sous-arbres.