INPG **ENSIMAG**
Année Spéciale **Année 2005-06**

Compilation

TP1 - Utilisation des outils Lex et Yacc

Le répertoire `bool.calc` contient un programme de *calculette booléenne*. Le programme prend en entrée une séquence de caractères représentant une expression booléenne et vérifie que la syntaxe de cette séquence est correcte, puis calcule et affiche sa valeur (true ou false). Ce programme a été écrit à l'aide des outils Lex et Yacc.

Répondez par écrit au questions suivantes sur la feuille de compte-rendu qui vous a été distribuée.

1 Observer

1. Copiez l'ensemble des fichiers du répertoire `bool.calc` dans un répertoire de votre choix.
2. Examinez les fichiers suivants :
   – `bool.l` : quels sont les lexèmes autorisés ?
   – `bool.y` : quelles sont les règles de la grammaire qui définissent une expression booléenne ? Dans cette grammaire l'opérateur $+$ est-il plus ou moins prioritaire que l'opérateur $\times$?
   – `main.c` : programme principal.
3. Compilez en exécutant la commande `make`.
4. Examinez les fichiers produits, notamment :
   – `y.output` : il s'agit d'un fichier de trace décrivant les tables de l'analyseur LALR(1) construit par Yacc. C'est l'équivalent de la table de transition de l'automate que nous construisions à la main en TD, par exemple :

```
state 5              /* état 5 de l'automate */
   F :  (_E )        /* l'unique item qui définit cet etat est "F -> (.E) */

   True  shift 6     /* lecture du lexème "True" et passage à l'état 6 */
   False  shift 7    /* lecture du lexème "False" et passage à l'état 7 */
   (  shift 5        /* lecture du lexème "(" et passage a l'état 5 */
   .  error

   E  goto 10        /* si un E est généré en sommet de pile on passe à l'état 10 */
   T  goto 3         /* si un T est généré en sommet de pile on passe à l'état 3 */
   F  goto 4         /* si un F est généré en sommet de pile on passe à l'état 4 */
```

   – `bool.c` : l'analyseur LALR(1) construit par Yacc en langage C.
5. Exécutez le programme `bool` en essayant différentes expressions booléennes, correctes ou non.

2 Nouvel opérateur

1. Modifiez les fichiers `bool.l` et `bool.y` pour ajouter à la grammaire un opérateur not de négation. Cet opérateur devra être plus prioritaire que tous les autres.
2. Compilez et testez le nouveau programme obtenu.

3 Conditionnelles

1. Modifiez les fichiers `bool.l` et `bool.y` pour ajouter à la grammaire un opérateur if *e1* then *e2* else *e3*. Cet opérateur prend en argument trois expressions booléennes, renvoie une expression booléenne (*e2* lorsque *e1* est vraie, *e3* sinon), et est moins prioritaire que tous les autres.
2. Compilez et testez le nouveau programme obtenu.
3. Ajoutez maintenant un opérateur if *e1* then *e2* qui prend en argument deux expressions booléennes et renvoie une expression booléenne (*e2* lorsque *e1* est vraie, false sinon).
4. Compilez. Que se passe t'il ? D'où vient le problème ?
5. Corrigez le problème en modifiant la syntaxe des opérateurs if *e1* then *e2* else *e3* et if *e1* then *e2* .