Maîtrise d'informatique, Analyse Syntaxique et Compilation.

Sujet de programmation en Lex

A remettre les 15 et 17 Novembre.

On vous demande d'écrire un programme Lex qui met au pluriel certaine phrases très simples du français.

On se contentera de phrases dont la forme est article nom verbe du premier groupe complément, ou pronom personnel verbe du premier groupe complément. La partie complément ne sera pas mise au pluriel.

Les exemples ayant le style suivant doivent être réalisés :

Le chien mange du pain et des viandes.

Les chiens mangent du pain et des viandes.

Un animal marche dans la forêt.

Des animaux marchent dans la forêt.

Je souhaite du silence.

Nous souhaitons du silence.

On vous demande de remettre un document qui contient le source Lex **écrit de façon simple et lisible**, les fonctions C seront limitées au strict minimum. Le document remis devra contenir aussi des exemples d'utilisation de votre programme dont certains donneront des mises au pluriel valables et d'autres erronées, de façon à montrer les limites de votre programme.

La réalisation de ce qui est strictement demandé (pluriel des articles, des verbes du premier groupe, des noms se terminant par al et des pronoms personnels) donnera droit à une note voisine de 12 sur 20. Les extensions personnelles vous apporteront des points supplémentaires.

Utilisation d'états en Lex

Pour faciliter la réalisation d'un traducteur en Lex on dispose de la notion d'*états* qui n'a pas été décrite en cours et que nous nous proposons d'introduire ici. Dans un programme Lex on peut utiliser des états afin de définir des règles qui ne s'appliquent que dans certaines situations.

On commence par déclarer les états possibles dans la première partie du programme (avant le premier symbole `%%`) dans des lignes commençant par `%x`.

Par exemple

```
%x APAIR BPAIR ABIMPAIR
```

Décrit trois états en plus de l'état `INITIAL` qui est implicitement existant dans tout programme Lex.

Ensuite une règle du programme (dans la seconde partie) doit être précédée par `<ETATx>`, et elle ne s'applique que si l'on se trouve dans l'état `ETATx`. Au départ on se trouve dans l'état `INITIAL` et pour changer d'état on effectue l'action `BEGIN(ETATx)` qui fait passer à l'etat `ETATx`.

Par exemple le programme suivant en Lex transforme les lettres c en $0, 1, 2, 3$ suivant qu'elles sont précédées d'un nombre pair de a et de b, ou d'un nombre impair de b et pair de a, ou d'un nombre impair de a et pair de b, ou d'un nombre impair de a et impair de b respectivement.

```
%option noyywrap
%x APAIR BPAIR ABIMPAIR

%%
```

```
<INITIAL>c    {printf ("0");}
<INITIAL>a {printf("a");BEGIN(BPAIR);}
<INITIAL>b {printf("b");BEGIN(APAIR);}

<APAIR>c    {printf ("1");}
<APAIR>a    {printf("a");BEGIN(ABIMPAIR);}
<APAIR>b   {printf("b");BEGIN(INITIAL);}

<BPAIR>c    {printf ("2");}
<BPAIR>b    {printf("b");BEGIN(ABIMPAIR);}
<BPAIR>a   {printf("a");BEGIN(INITIAL);}

<ABIMPAIR>c    {printf ("3");}
<ABIMPAIR>a    {printf("a");BEGIN(APAIR);}
<ABIMPAIR>b   {printf("b");BEGIN(BPAIR);}

%%
main() {
      yylex();
      }
```