TP 1 : Introduction à Lex

1 Écriture d'expressions régulières

Ecrire les expressions régulières permettant de reconnaître les langages suivants, les programmer en OCAMLLex et les tester.

On pourra utiliser le petit script suivant :

```
rule main = parse
        expression_reguliere { print_string "SUCCES !\n" }
|       _                    { print_string "ECHEC !\n" }

{
main (Lexing.from_channel stdin)
}
```

Compiler avec la commande :

```
> ocamllex programme.mll
> ocamlc -o programme programme.ml
```

Et réaliser le test de la manière suivante :

```
> ./programme
mot_a_tester
```

Les langages à traduire sont les suivants :

1. Les nombres naturels : 0, 666, 123.
2. Les nombres entiers signés : -5, 0, 55, +33.
3. Les mots “on” et “off” où n'importe quelle lettre peut être une majuscule ou une minuscule.
4. Les nombres hexadécimaux non-signés : 123, f4, ffff, abcd, 4ff0.
5. Les nombres flottants : +123.5, 0.5e23, -12.66E-5.
6. Les chaînes de caractère en C : ””, ”abcd f”, ”ab\”cd\\ef”.
7. Les commentaires en C : /* abc */, /*** *** ***/.
8. *(optionnel) :* Les heures exprimées sous les formes : 8 :32, 14 :55, 09 :00 :26.
9. Les mots formés de parenthèses bien appareillées “((()(()))”.

2 Traducteur de HTML en LaTeX

Nous allons écrire à l'aide de Lex un traducteur de HTML en LaTeX. HTML est un langage structuré par des balises contenant des méta-informations (structure du document, mise en page, sémantique du texte, etc). Les balises débutent par le caractère < et se terminent par le caractère > et peuvent contenir n'importe quel texte excepté > et <. Les balises peuvent définir des zones de texte et, dans ce cas, la balise de fermeture de la zone débute par le caractère /. Voici un exemple de fichier HTML :

```
<html>
<body>
Hello, <b>World</b> !<br>
Hello, <i>World</i> !
</body>
</html>
```

Un document LaTeX est également un format utilisant des balises mais sous une forme différente. [1] Par exemple, le fichier HTML ci-dessus peut se traduire en LaTeX sous le forme suivante :

```
\documentclass{article}
\begin{document}
Hello, \textbf{World} !
\newline
Hello, \emph{World} !
\end{document}
```

Cet exercice consiste à écrire, en utilisant Lex, un traducteur d'HTML en LaTeX qui prendra le document HTML sur l'entrée standard et produira le fichier en LaTeX sur la sortie standard. On pourra tester le fonctionnement du traducteur par les commandes suivantes :

```
> traducteur < fichier.html > fichier.tex
> latex fichier.tex
> xdvi fichier.dvi
```

On utilisera les correspondances suivantes entre les structures des deux langages :

<html>...</html>	\documentclass{article}...
<body>...</body>	\begin{document}...\end{document}
...	\section{...}
...	\subsection{...}
...	\begin{enumerate}...\end{enumerate}
...	\begin{itemize}...\end{itemize}
 	\newline
...	*(ligne vide)*...
...	\textbf{...}
...	\emph{...}
...	\item ...

On traduit
 par \newline uniquement si
 ne se trouve pas au début de la ligne. Si une ligne commence avec un
, on le traite comme .

3 Calculatrice en polonaise inversée *(optionnel)*

Il s'agit de réaliser ici une petite calculatrice en notation polonaise inversée. Je rappelle qu'en notation polonaise inversée, on trouve d'abord les opérandes d'un calcul puis l'opérateur. Par exemple, le calcul $3 \times (4 - 1)$ devient 3 4 1 $-$ $\times$. On réalise ce calcul en utilisant une pile. Si un nombre est trouvé, il est empilé. S'il s'agit d'un opérateur, il s'applique sur les opérandes en sommet de pile qu'il remplace par le résultat du calcul. Dans notre exemple, on empile 3, 4 et 1, [3, 4, 1], puis on effectue la soustraction, [3, 3], et enfin la multiplication, [9]. Il faut remarquer que la notation polonaise inversée ne nécessite pas

1. Ce document-ci a été écrit en LaTeX. Voir https://www.latex-project.org/

de parenthèse : il n'y aucune ambiguité dans la notation. Cependant, elle est plus difficile à lire pour les humains.

Dans cet exercice, il est demandé d'utiliser Lex pour réaliser notre calculatrice en polonaise inversée. On implante d'abord la calculatrice simple : les nombres sont de type flottant (notation standard) et on implante les opérations `+`, `-`, `*`, `/`, `**` (exposant), `--` (négation unaire). La calculatrice accepte plusieurs calculs, chacun terminés par "`;`", et l'ensemble terminé par un "`.`".