**Talal Jawaid**

**5/12/2019**

**CSC 135**

**PL4**

<body>

<h1>Talal Jawaid</h1>

<table border="0" cellpadding="0" cellspacing="0" style="border-collapse: collapse" bordercolor="#111111" width="100%" id="AutoNumber1">

<tr>

<td width="9%">&nbsp;</td>

<td width="79%"><h2><font size="5" color="#800000">PL Homework Assignment 4: </font></h2>

<p><font size="4"><b>Web-based Recursive Descent Recognizer</b></font></p>

<p class="MsoNormal">

<font size="4">In this assignment you are to implement a recursive-descent

recognizer with a<strong> web interface</strong> for the BNF grammar&nbsp;given below. Based on the pseudocode you have done in PL Assignment 1, this is a good opportunity to develop the web programming skills required by today's IT field.</font></p>

<p class="MsoNormal"><font face="Times New Roman" size="4">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; EXP&nbsp;&nbsp;&nbsp; ::= EXP&nbsp; + TERM&nbsp;&nbsp; | EXP - TERM&nbsp;&nbsp;&nbsp; | TERM<br>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; TERM&nbsp;&nbsp; ::= TERM \* FACTOR | TERM / FACTOR | FACTOR<br>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; FACTOR ::= ( EXP ) | DIGIT<br>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; DIGIT&nbsp; ::= 0 | 1 | 2 | 3</font></p>

<p class="MsoNormal"><font size="4">The following is a set of requirements for this recognizer (parser):</font></p>

<p class="MsoNormal"><font size="4">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; - Ask the user for an input stream.<br>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; - Report &quot;legal&quot; or &quot;errors found&quot; (not both!).<br>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; - Assume the input stream is the token stream.<br>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; - Assume the input stream terminates with a $.<br>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; - Assume there is no white space.<br>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; - Use a form to collect input and return the output.<br>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; - Test your recognizer with illegal

and legal strings.<br>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; - Give a brief description of this recognizer for the user on how to

use and what method that it based on.</font></p>

<p class="MsoNormal"><font size="4"><b>Submission:</b> </font>

<p class="MsoNormal"><font size="4">1. <b>Test</b> your RDR thoroughly.</font> <font size="4">Based on the given grammar, you may construct your testing strings with a set of valid ones and a set of invalid ones. Here is two sample testing sets:</font>

<p class="MsoNormal"><font size="4">Valid set: 1+2$ , 0-3\*2$, 2/(3+1)$ , 1+2+3+0$, 3+2\*3/2$, (0+2)/(((0+3)-2)/2)$, (((1+3)\*2)-3)$ </font>

<p class="MsoNormal"><font size="4">invalid set: 1+2, 1\*4$,((1\*2)/3$ , 2\*$ , (1+a)$ , 1+1 </font> 1\*a$

<p class="MsoNormal">

<p>

<p><h2><a href="http://athena.ecs.csus.edu/~jawaidt/csc135/parser.docx">Parser Report </a></h2></p>

<p>

<h4>Input String: </h4>

<input type="text" id="input" >

<button onclick="submit()"> Submit </button> </br></br>

<p>

<script>

var inputString;

var index1;

var errorflag;

function submit()

{

index1=0;

errorflag=0;

inputString = document.getElementById("input").value;

if(inputString[inputString.length-1]==="$")

Start\_scan();

else

confirm("Not a valid string! $ sign is missing. ");

};

function Digit()

{

if ((Token() === '0') || (Token() === '1') || (Token() === '2') || (Token() === '3') )

validate\_token(Token());

else

Error();

};

function Factor()

{

if(Token()==='(')

{

validate\_token(Token());

Expr();

validate\_token(')');

}

else if ((Token() ==='0') || (Token() === '1') || (Token() === '2') || (Token() === '3'))

{

Digit();

}

else

Error();

};

function Term()

{

Factor();

while((Token()==='\*') || (Token()==='/'))

{

if(Token()==='\*')

{

validate\_token(Token());

Factor();

}

else if(Token()==='/')

{

validate\_token(Token());

Factor();

}

else

Error();

}

};

function Token()

{

return(inputString[index1]);

};

function Inc\_pointer()

{

if (index1 < (inputString.length -1))

index1++;

};

function validate\_token(x)

{

if (x === Token())

Inc\_pointer();

else

Error();

};

function Expr()

{

Term();

while((Token()==='+') || (Token()==='-'))

{

if(Token()==='+')

{

validate\_token(Token());

Term();

}

else if(Token()==='-')

{

validate\_token(Token());

Term();

}

else

{

Error();

}

}

};

function Error()

{

confirm("Invalid String! :( Error at position: " + (index1 + 1));

errorflag = 1;

Inc\_pointer();

};

function Start\_scan()

{

Expr();

validate\_token('$');

if (errorflag === 0){

confirm("Valid String! :)" + "\n");

}

};

</script>