Compiler

Lab 4: Jflex with scanner (Lexical analysis)

Exercise 1

Write a scanner for the C programming language. The scanner must:

- Recognize the C language comments (/* and */)
- Recognize the symbols: “{“, “}”, “(“, “)”, “[“, “]”, “+”, “-”, “*”, “/”, “=”, “;”, “.”, “,”, “<”, “>”, “&”, “|”, “!”
- Recognize some C keywords: int, double, if, else, while, print
- Recognize integer and double numbers
- Any #include, space, tab and newline must be discarded

The scanner must provide as output the recognized units (the TOKENS) printing them separated by a space. Regarding the integers, the double numbers and the identifiers, it must also print, in addition to the recognized token, the token value (e.g. INT:30)

1.1 Input file example

Observe the example. Given the following input file:

```
  double x[5];
  int i, j; double swap;
  int pos;
  /* Vector initialization */
  x[0] = -2.0;
  x[1] = -3.0;
  x[2] = 3.0;
  x[3] = 5.0;
  x[4] = 2.5;
  /* Bubble sort */
  pos = 5;
  while(pos > 0){
      i = 0;
      while (i < pos - 1){
          j = i + 1;
          if (x[i] > x[j]){
                  swap = x[j];
                  x[j] = x[i];
                  x[i] = swap;
           }
           i = i + 1;
      }
      pos = pos-1;
}
```

```
/* Print results */
i = 0;
while(i<5){
      print (x[i]);
      i = i + 1;
}
```

1.2 Output Example

the output must be the following:

```
DOUBLE_TYPE ID:x BO INT:5 BC SCL INT_TYPE ID:i COM ID:j SCL
DOUBLE_TYPE ID:swap SCL INT_TYPE ID:pos SCL ID:x BO INT:0 BC EQ
MIN DOUBLE:2.0 SCL ID:x BO INT:1 BC EQ MIN DOUBLE:3.0 SCL ID:x BO
INT:2 BC EQ DOUBLE:3.0 SCL ID:x BO INT:3 BC EQ DOUBLE:5.0 SCL ID:x
BO INT:4 BC EQ DOUBLE:2.5 SCL ID:pos EQ INT:5 SCL WHILE PO ID:pos
BIG INT:0 PC CBO ID:i EQ INT:0 SCL WHILE PO ID:i SML ID:pos MIN INT:1
PC CBO ID:j EQ ID:i PLUS INT:1 SCL IF PO ID:x BO ID:i BC BIG ID:x BO ID:j
BC PC CBO ID:swap EQ ID:x BO ID:j BC SCL ID:x BO ID:j BC EQ ID:x BO
ID:i BC SCL ID:x BO ID:i BC EQ ID:swap SCL CBC ID:i EQ ID:i PLUS INT:1
SCL CBC ID:pos EQ ID:pos MIN INT:1 SCL CBC ID:i EQ INT:0 SCL WHILE
PO ID:i SML INT:5 PC CBO PRINT PO ID:x BO ID:i BC PC SCL ID:i EQ ID:i
PLUS INT:1 SCL CBC
```

Exercise 2

Write a lexical analyzer using JFLEX which is able to recognize the principal elements of an HTML document. An HTML document consists of an ASCII text annotated by appropriate keywords.

All the keyword are enclosed between the symbols “<” and “>”. Within these symbols there could also be some modifiers and keyword parameters. Keywords and parameters are case insensitive. The two delimiter characters and their content define a tag. The keywords consist of alphanumeric characters and begin with an alphabetic character. Furthermore, the closing tags can be preceded by the character “/”. An HTML document can contain some comments. Comments start with characters “<!−−” and end with characters “−− >”. Among the keywords that could appear, the lexical analyzer must explicitly recognize the keywords “head”, “body”, “html”, “title”, “table”, “h1”, “h2”, “h3”, “h4”.

The lexical analyzer must produce as output the HTML input document with all comments removed. Furthermore, at the end of the file it must print the following statistics:

- the total number of tags
- the number of table, h1, h2, h3 and h4 tags

2.1 Input file example

```
<HTML><HEAD><TITLE>Proof</TITLE></HEAD>
<BODY>
<!-- .... <table>A table inside a comment (not count)</table> -->
<H1>Title_1</h1>
<h2>Title_1_1</H2>
```

```
Test <b>various</b>
<H1>Title_2</h1>
<h2>Title_2_1</H2>
<table border=2><tr><td>Idem</td></tr></table>
<a href="top.html"><img src="/img/top.gif" alt="top"></a>
<h2>Title_2_2</H2>
<table border=0><tr><td>
<table border=0><tr><td> Annexed Table first level</td></tr>
</table>
</table>
<!-- unordered list tag -->
<ul>
<li><a href="pippo.htm">pippo</a>
<li><a href="pluto.htm">pluto<i> and </i>pippo</a>
</ul>
<table border=0><tr><td>
<table border=0><tr><td>
<table border=0><tr><td>Annexed Table second level</td></tr>
</table>
</td></tr>
</table>
</table>
<img src="/img/bottom.gif " height="10" width="10" alt="bottom">
<hr>
```

2.2 Output example

```
<HTML><HEAD><TITLE>Proof</TITLE></HEAD>
<BODY>
<H1>Title_1</h1>
<h2>Title_1_1</H2>
Test <b>various</b>
<H1>Title_2</h1>
<h2>Title_2_1</H2>
<table border=2><tr><td>Idem</td></tr></table>
<a href="top.html"><img src="/img/top.gif " alt="top"></a>
<h2>Title_2_2</H2>
<table border=0><tr><td>
<table border=0><tr><td>Annexed Table first level</td></tr>
</table>
</table>
<ul>
<li><a href="pippo.htm">pippo</a>
<li><a href="pluto.htm">pluto<i> and </i>pippo</a>
</ul>
<table border=0><tr><td>
<table border=0><tr><td>
<table border=0><tr><td>Annexed Table second level</td></tr>
</table>
</td></tr>
</table>
</table>
<img src="/img/bottom.gif " height="10" width="10" alt="bottom">
```

```
<hr>
</body></HTML>
Total number of tags: 67
Total number of table tags: 12
Total number of h1 tags: 4
Total number of h2 tags: 6
Total number of h3 tags: 0
Total number of h4 tags: 0
```