A Scanner for a Small JavaScript

Due Date: April 8th, Monday, 10 PM
Please, submit as early as possible.

Deliverables:
a set of **regular expressions**, (optional NFA), **DFA**, Java **source code**, **Design Document** with UML or any notations, a **snap-shop of running result** program with test inputs, and **User's Manual** for running your program.
Name: smalljs.java, which contains main method.
The zip file name format: YourName_Scanner.zip

Notice: When you turn in your working, make sure your program works in DOS command line for the Java source without using any tools such as Eclipse or any other development environment.
(제출하기 전에 반드시 javac 와 java 를 사용하여 command line 에서 test 한 후에 source code 를 제출해 주시길 바랍니다. source code 는 특정 IDE 에서 만든 directory 는 제출하지 마시길 바랍니다.)
Your first job is to **make regular expressions** for the token groups, which are user-defined identifiers, keywords, literals, special characters for punctuations, operators for comparisons, arithmetic operators, operator assignment, and so on.
For each regular expression group, try to make a DFA or a NFA.
Lastly convert DFA into real Java coding.
You have an option to recognize tag names inside string literals.
For example, in ("<h1>Examination Results</h1>") the string surrounded by a pair of double quotes can be regarded as string literal or optionally you can extract <h1> and </h1> as tag name keyword and get some extra credit. (double quote 안에 있는 string 에서 tag name 을 keyword 로 인식하는 것은 option 입니다. double quotes 안에 있는 것들은 모두 string literal 로 인식해도 됩니다. tag name 으로 인식하도록 만들면 extra credit 을 드립니다.)

Test Input: test1.jss

```
<script_start>
   //initialization
   var passes = 0, // number of passes
   student = 1, //student counter
   result;
   var number, sum = 0;
   var input1 = window.prompt("Enter the first number");
   var value1 = parseFloat(input);
   //process 10 students
   while (student <= 5) {
      result = window.prompt("Enter result");
      if (result = = "1")
           if ( result = = "2" ) {passes = passes + 1; square (result);}
           else
           failures++;
           ++student;
```

keyword inside < and

keyword idenfier

literal

```
}
// termination phase
document.writeln("<h1>Examination Results</h1>");
document.writeln("<h2>Passed and Failed Numbers </h2>");
if ( passes > 8 )
        document.writeln( "<br />Raise Tuition" );

for (number = 2; number <= 100; number++)
        sum += number;
        document.write( "The sum of the even integers from 2 to 100 is ");
        document.writeln(sum);

switch ( choice ) {
    case "1":
         startTag = "<ul>";
         endTag = "</ul>";
         break;
    case "2":
         startTag = "<ol>";
         endTag = "</ol>";
         listType = "<h3>Ordered List: Numbered</h3>";
         break;
    case "3":
         startTag = "<ol >";
         endTag = "</ol>";
         listType = "<h3>Ordered List: Lettered</h3>";
         break;
      default:
         validInput = false;
        }

        do {
           document.writeln();
             ++counter;
        } while ( counter <= 6 );
        function square(y)
        {
            return y*y;
        }
        <script_end>
```

keyword inside < and >

keyword idenfier

function name

Sample running result is given below:

```
java smalljs test1.jss
Scanning starts
<script_start>    keyword id
//initialization    comment
var       keyword id
passes    user-defined id
=         assignment operator
0         number
,         punctuation charcter
//number of passes        comment
student   user-defined id
…
windows.prompt          keyword or
windows.prompt          keyword function name
(“Enter the first number”)          function parameter or literal
…
parseFloat        keyword function name
(input)           function parameter
..
= =       equality comparison operator
..
++        increment operator
..
document.writeln          keword function name
..
<h1>      keyword tag name    //this is optional
…
</h1>     keyword tag name or keyword ending tag name //this is optional
…
<=        less than or equal comparison operator
…
+=        assignment operator
```

or you can simply strip off comment without displaying comment as program output