Development of the LR(1) Parser for SmallJS

1st Due Date: June 15th, 10 pm, Saturday
2nd Due Date: June 19th, 10 pm, Wednesday
Deliverables: The file containing main file, "LR1.java," and all the aiding java files, BNF, **First Set, Follow Set, DFA of LR(1) Items, LR(1) Parsing Table**, User's Manual for running your program, test inputs, Optional Developer's Note, Required Screen Shots Showing the Running of Your Program
For option 2, submit .l file, .y file, user's manual, test inputs, and the screen shots of running results.
The zip file containing all the deliverables should be named as **English_Name_LR1.zip.**
When you use MaC PC, do not include Korean character into the zip file name.

Submit early for more credit.
Remove the statements, *package myPackage* that may be in the first line of the source code.
Test your source code in DOS command line before submitting your workings.
zip file 에 한글 이름 넣지 마시길 바랍니다.

Description (Option 1): Try to make a LR(1) parser for the test language, which can check the validity of the grammar of sample input programs.
This project is to make a LR(1) parser.
Prepare the First Set and Follow Set from the your BNF, and make the DFA of LR(1) items. Then make a LR(1) parsing table. Implement as much as you can.

Description (Option 2): For option 2, the full credit will be 70%. You may try to make a parser using Flex and Bison. Try to remove all the conflicts generated by Bison.

Grammar and Spelling Rules

1. User-defined identifier starts with an alphabet letter followed by alphabet or number.
2. The program is case-sensitive.
3. Comments start with "//" until the end of current line.
4. The assignment is '=', and the equality symbol is '=='.
5. Program starts with <script_start> and ends with <script_end>.
6. The semicolon is a statement terminator.
7. The parser does not consider the tag names of HTML.

Notice!

When your parser encounters an illegal identifier or incorrect sentence, the parsing process just stops.

You do not need to implement calculation. Just check grammar.

The following is an example output of the parser. Your output may be slightly different.

```
//Basic Test Input 1//
<script_start>
var temperature = 20, change = 0;
var limit = 40, fan = 0;
temperature = temperature + change;
limit = limit + fan;
<script_end>
```

java LR1 a1.jss

parsing OK

```
////////
//Basic Test Input 2//
<script_start>
var temperature = 20, change = 0;
var limit = 40, fan = 0;
temperature = temperature + change;
limit = limit + fan - ;

<script_end>
```

java LR1 a1Error.jss

parsing error
No operand at fan -


```
////////////////////////////////////////////////////////////////
<script_start>
var temperature = 20, change = 0;
var limit = 40, fan = 0;
temperature = temperature + change;
limit = limit + fan;
if (temperature > limit)
   { temperature = 20;
      limit = limit – 5;
   }
else
   { temperature = temperature + 10;
      limit = limit + 5;
   }
<script_end>
```

java LR1 basic1.jss

parsing OK

```
/////////
//Basic Test Input 2//

<script_start>
var temperature = 20, change = 0;
var limit = 40, fan = 0;
temperature = temperature + change;
limit = limit + fan ;
<script_end>
```

java LR1 basic2.jss

```
parsing error
temperature3 undefined

//sample1.jss is valid
<script_start>
var temperature = 20;
var limit = 40, fan = 0;

// temperature monitoring
while (temperature <= limit) {

   if (temperature == limit) {
      document.writeln("temperature limit");
      temperature = 20;
      fan = 1;
   }
   else
   {
      temperature++;
      fan = 0;
   }
}
<script_end>
```

java LR1 sample1.jss

```
parsing OK

//sample2.jss is invalid
<script_start>
var 9temperature = 20;
var limit = 40, fan = 0;
while (temperature <= limit) {
   if (temperature == limit) {
      document.writeln("temperature limit");
      temperature = 20;
      fan = 1;
```

Test illegal syntax.

- missing parenthesis at if and else structure
- missing <script_end>

```
    }
    elsa {
        temperature++;
        fan = 0;
    }
}
<script_end>
```

The error message is up to you.

java LR1 sample2.jss

```
parsing error
elsa is invalid keyword…
```

```
//sample3.jss
<script_start>
var temperature = 20;
var limit = 40, fan = 0;
var initial = 0;
var accumulation;

for (temperature = initial; temperature < limit; temperature++) {
      document.writeln("normal temperature");
      accumulation = accumulation + 5;
}

<script_end>
```

java LR1 sample3.jss

```
parsing OK
```

```
//sample4.jss is invalid
<script_start>
var temperature = 20;
var limit = 40, fan = 0;
var initial = 0;
var accumulation;
```

Test the three components of for statement.
Missing a part of for statement…

```
for (temperature = initial; temperature <    ; temperature++) {
      document.writeln("normal temperature");
      accumulation = accumulation + 5;
}
<script_end>
```

java LR1 sample4.jss

```
parsing error
temperature <
```

```
//sample5.jss
<script_start>
var temperature = 20;
var limit = 40, fan = 0;
var choice = 0;
var accumulation;

choice = 1;
switch ( choice ) {
   case "1":
     limit = 10;
     fan = 5;
     break;
   case "2":
     limit = 20;
     fan = 10;
     break;
   case "3":
     limit = 30;
     fan = 20;
     break;
 default:
   limit = 40;
 }
<script_end>
```

java LR1 sample5.jss

```
parsing OK
```

```
//sample6.jss is invalid
<script_start>
var temperature = 20;
var limit = 40, fan = 0;
var choice = 0;
var accumulation;

choice = 1;
switch ( choice ) {
   sase "1":
     limit = 10;
     fan = 5;
     break;
   case "2":
     limit = 20;
     fan = 10;
     break;
```

Test the case statement for various invalid keywords.

```
    case "3":
      limit = 30;
      fan = 20;
      break;
  default:
    limit = 40;
  }
<script_end>
```

java LR1 sample6.jss

```
parsing error
  sase "1":
```