Simple JSON Parser

Deadline： 23:59, June. 12th, 2017

1. Description

In this assignment, you need to write a simple JSON parser which can complete the analysis of JSON text and extract specific information from it.

Your submitted codes will be checked by MOSS for plagiarizing. You will get ***ZERO***, if a high repetition rate has been found.

2. Requirements

2.1 Rigidity requirements (50pts)

a. You might complete this assignment all by yourself, if it seems to be very hard for you, you can also team up with another classmate and submit just one copy of your works;
   If a high repetition rate has been found, all of them will get ZERO.
b. Submit your works before deadline;
   If you can't finish it before deadline, your final score will be reduced by 10pts per day until you get ZERO.
c. You should code with C/C++ or java, no third party libraries are expected, except of standard libraries.

2.2 Functionality requirements (40pts)

All test cases are encoded with UTF-8.

Input 1:

```
{
    "id": 2333,
    "name": "compiler"
}
```

Output 1:

```
valid
```

Input 2:

```
{
    "id": 2333
```

```
    "name": "compiler"
}
```

Output 2:

line 2, position 12: expected <,>

For abcd below, if there are no errors while parsing JSON text, just output “valid”; otherwise, point out where errors are (line number and start position) and why they happened.

a. Parsing JSON’s basic structure, includes key-value pair, object, array, string;(10pts)
b. Parsing ”true”, “false”, ”null”, integer; (5pts)
c. Parsing floating point number; (5pts)
d. Parsing scientific notation; (5pts)
e. Formatting JSON text to a pretty format. Add a command argument “-pretty” to your program, if there are no errors after parsing file “*.json”, output a formatted pretty JSON text to another file named “*.pretty.json”; (5pts)

   You can find a disordered JSON text named “country.json” and a pretty JSON text named “country.pretty.json” in directory “test/e/”.

   For example, your program is named “json”, so commands should be like this:

```
json -pretty *.json
```

f. For any JSON text and a given query path, you can find the specified value and determine the type of that value (string? floating point numbers? object? array? etc.). (10pts)

   For example, for the supplied file country.json, let us suppose that the query path is “/RECORDS[35]/countryname”, so it means extracting the value of the “countryname” field of the 35th object of the array corresponding to the “RECORDS” field, that is, a string "China"; if it doesn’t exist, just return null.

   You are free to design the representation of query path.

   JSON itself is just like a tree structure, so in this part, what you should focus on is how to express the query path and how to query the required information, you can refer to XML’s parsing (XPath), but there is no need to be exactly the same.

2.3 Structure requirements(10 pts)

a. Your program should follow the structure of a parser, read the source file -> lexical analysis -> grammar analysis -> simple semantic analysis -> implementation of semantics (5pts)
b. Modular, reusable code design. (5pts)

3. JSON

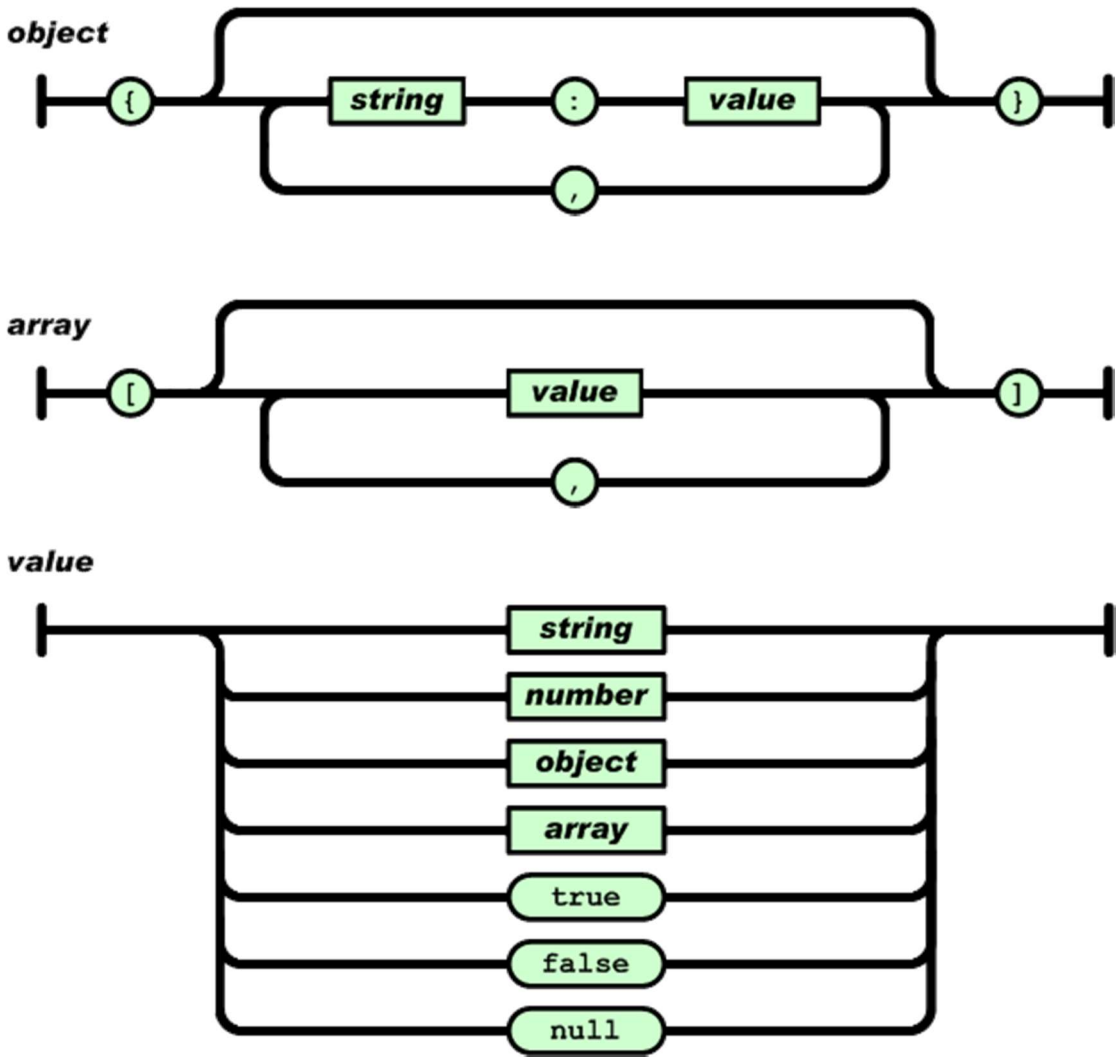
3.1 BNF

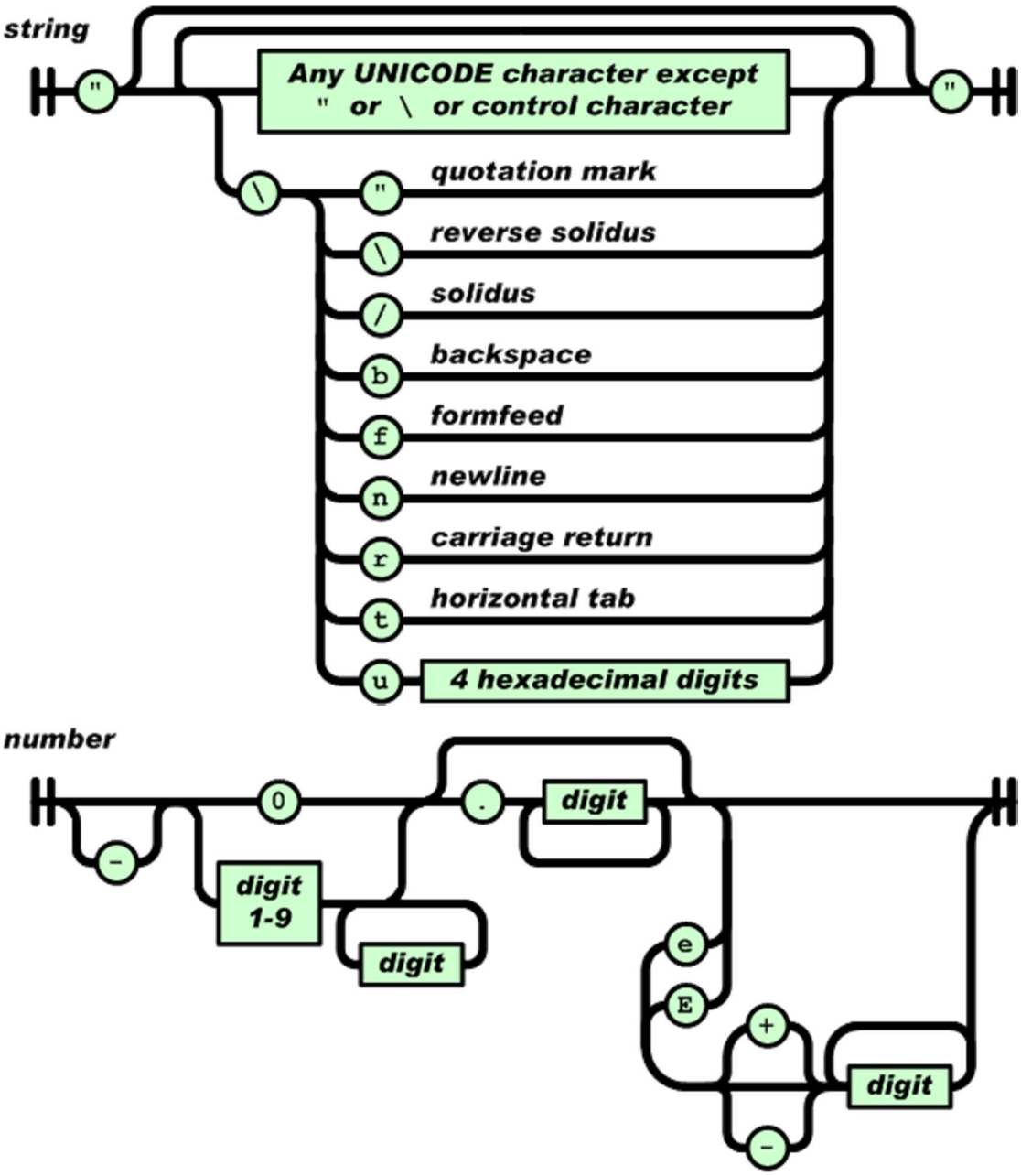
```
<json> ::= <object> | <array>
<object> ::= "{" "}" | "{" <members> "}"
<members> ::= <pair> | <pair> "," <members>
<pair> ::= <string> ":" <value>
```

<array> ::= "[" "]" | "[" <elements> "]"
<elements> ::= <value> | <value> "," <elements>
<value> ::= <string> | <number> | <object> | <array> | "true" | "false" | "null"
<number> ::= <integer> | <float> | <scientific>

Among them:
<string>: A string is a sequence of zero or more Unicode characters, wrapped in double quotes, using backslash escapes
<integer>: Integer number
<float>: Float point number
<scientific>: Scientific notation
Whitespace can be inserted between any pair of tokens.

3.2 DFA





Reference: http://json.org

4. Submission

1) Compile your code to an executable file, .jar or .exe.
2) No need for GUI, and your executable file should accept a source file path as command arguments.
3) Executing without errors, your program should exit immediately, DO NOT put cin.get() or system("pause") into you code to wait keys.
4) Write a document describing your implementation; Screen captures of running result is required; If you team up with another classmate, then both of you should write one and illustrate what you have done.
5) Source codes should be put into a directory named "src", executable file and your

document have the same level with directory “src”.

6) Compress your homework into a .zip file, named in this format, “StudentID-class-name”, or “StudentID1&StudentID2-class-name1&name2” for teams.
7) Send your homework to compiler2017@126.com before 23:59, June. 12th, 2017.