Parser conflicts + JSON parser

Lab assignment

In this lab we will observe RR and SR conflicts + write a JSON parser

JSON

Notice the recursive structure!

```
{
  "glossary": {
    "title": "example glossary",
    "GlossDiv": {
      "title": "S",
      "GlossList": {
        "GlossEntry": {
          "ID": "SGML",
          "SortAs": "SGML",
          "GlossTerm": "Standard Generalized Markup Language",
          "Acronym": "SGML",
          "Abbrev": "ISO 8879:1986",
          "GlossDef": {
            "para": "A meta-markup language, used to create markup languages such as DocBook.",
            "GlossSeeAlso": [
              "GML",
              "XML"
            ]
          },
          "GlossSee": "markup"
        }
      }
    }
  }
}
```

LR(1) parser

"1" is referring to 1 lookahead symbol. It is a data being passed along while trainsitioning the parser states.

- Basically we can reduce if the lookahead symbol is the symbol we intended it to be.
- lookahead symbol b in production $A \to \cdot Bc, b$ should be $\in \text{FOLLOW}(A)$

SR non-terminals

```
header
nullable: No
firsts: IDENTIFIER
follows: $end
Productions: 1 2

params
nullable: No
firsts: IDENTIFIER
follows: ),,
Productions: 3 4

param
nullable: No
firsts: IDENTIFIER
follows: ),,
Productions: 5

ids
nullable: No
firsts: IDENTIFIER
follows: ,,)
Productions: 6 7

type
nullable: No
firsts: IDENTIFIER
follows: IDENTIFIER
Productions: 8

id
nullable: No
firsts: IDENTIFIER
follows: (,),,
Productions: 9

$accept
nullable: No
firsts: IDENTIFIER
follows:
Productions: 0
```

SR LR(1) productions

```
0)   $accept -> header $end
1)   header -> type id ( params )
2)   header -> type id ( )
3)   params -> param
4)   params -> params , param
5)   param -> type ids
6)   ids -> id
7)   ids -> ids , id
8)   type -> IDENTIFIER
9)   id -> IDENTIFIER
```

SR LR(1) parse table

↓states	$end	(	)	,	IDENTIFIER	header	id	ids	param	params	type
0					s3	1					2
1	a										
2					s5		4				
3					r8						
4		s6									
5		r9									
6			s8		s3				9	7	10
7			s11	s12							
8	r2										
9			r3	r3							
10					s15		14	13			
11	r1										
12					s3				16		10
13			r5	s17,r5 - Shift , then Go to state 17 - Reduce by 5) param -> type ids							
14			r6	r6							
15			r9	r9							
16			r4	r4							
17					s15		18				
18			r7	r7							

RR LR(1) productions

```
0)   $accept -> exp $end
1)   exp -> exp - sub_exp
2)   exp -> sub_exp
3)   sub_exp -> ( type ) sub_exp
4)   sub_exp -> - sub_exp
5)   sub_exp -> IDENTIFIER
6)   sub_exp -> ( exp )
7)   type -> IDENTIFIER
```

RR LR1(1) non-terminals

```
exp
nullable: No
firsts: (,-,IDENTIFIER
follows: $end,-,)
Productions: 1 2

sub_exp
nullable: No
firsts: (,-,IDENTIFIER
follows: -,$end,)
Productions: 3 4 5 6

type
nullable: No
firsts: IDENTIFIER
follows: )
Productions: 7

$accept
nullable: No
firsts: (,-,IDENTIFIER
follows:
Productions: 0
```

RR LR(1) parse table

↓states	$end	(	)	-	IDENTIFIER	exp	sub_exp	type
0		s3		s4	s5	1	2	
1	a			s6				
2	r2			r2				
3		s11		s12	s9	8	10	7
4		s3		s4	s5		13	
5	r5			r5				
6		s3		s4	s5		14	
7			s15					
8			s16	s17				
9			r7,r5 - Reduce by 7) type -> IDENTIFIER - Reduce by 5) sub_exp -> IDENTIFIER	r5				
10			r2	r2				
11		s11		s12	s9	19	10	18
12		s11		s12	s21		20	
13	r4			r4				
14	r1			r1				
15		s3		s4	s5		22	
16	r6			r6				
17		s11		s12	s21		23	
18			s24					
19			s25	s17				
20			r4	r4				
21			r5	r5				
22	r3			r3				
23			r1	r1				
24		s11		s12	s21		26	
25			r6	r6				
26			r3	r3				