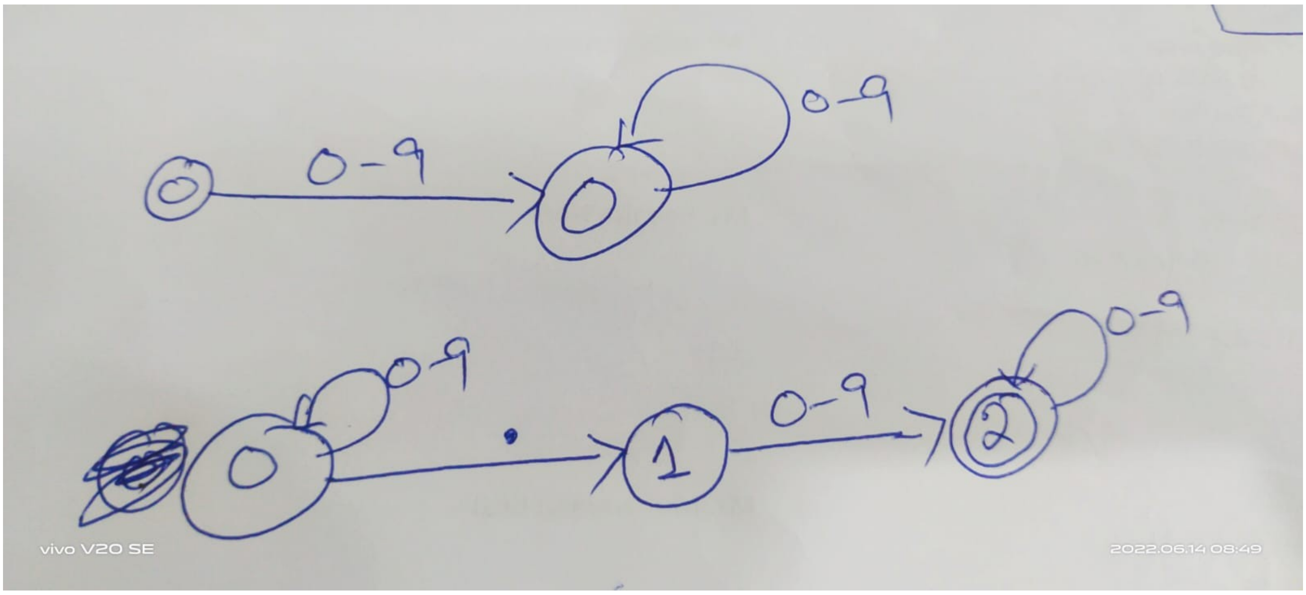
Q. Consider the following grammar.

E→ T print{“ E-T is reduced”}
E → E + T print{“E + T is reduce”}
T → F print{“F is reduce”}
T → T * F print{“ T*F reduce”}
F → a print{ “a is reduce , value is” a.val}
F → b print{ “b is reduce , value is” bs.val}
F → (E)

consider ‘a’ as integer and ‘b’ as float number. Following are the NFA for integer and float number.



Following are the table for integer and float NFA.

Present State	[0-9]
0	1
1	1

For float

Present State	[0-9]	.
0	0	1
1	2	-
2	2	-

Following is LALR parsing table for this grammar.

LR table

State	ACTION							GOTO		
	+	*	a	b	(	)	$	E	T	F
0			s3	s4	s5				1	2
1	acc	s6				acc	acc			
2	r2	r2				r2	r2			
3	r4	r4				r4	r4			
4	r5	r5				r5	r5			
5			s3	s4	s5			7	1	2
6			s3	s4	s5					8
7	s10					s9				
8	r3	r3				r3	r3			
9	r6	r6				r6	r6			
10			s3	s4	s5				11	2
11	r1	s6				r1				

→ You are required to do following tasks.

1. Save all these tables in files , you can name as per your choice and can also use any pattern for table.

2. You need to read input from input.txt and identify tokens. (For example for input 35 + 45.5 tokens will be a+b.)

3. You need to parse identified tokens using LALR-1 parser. And each action should be printed (either it is shift, reduce etc).

4. You also need to execute corresponding semantic rules against each reduction.