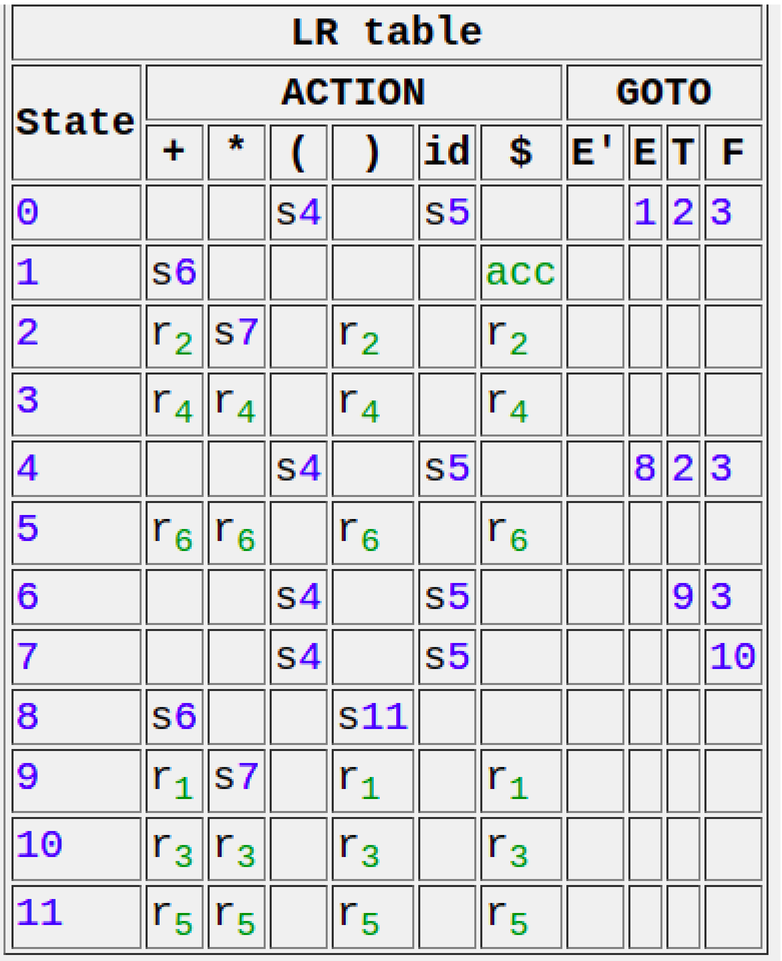
C++ has many type of tokens, including operators,precedence and associativity. Following are few well known operators and associativity.

::   scope resolution
[] array subscript
++ postfix increment
- - postfix decrement
~ one,s complement
! logical NOT
* Multiplication operators
< Less then
> greater then
<= less then or equal to
>= greater then are equal to
!= not equal to
>>= Right shift Assignment
^=   Bitwise XOR Assignment

Consider the following SDT
E' -> E   {print(“At last Accepted”)}
E -> E + T   {print(“T is reduced “)}
E -> T   {print(“ E+T is reduced”)}
T -> T * F   {print(“ F is reduced”)}
T -> F   {print(“ T*F is reduced”)}
F -> (E)   {print(“ (E) is reduced”)}
F -> id   {print(id.val , “ is reduced”)}

where id hold the properties of a identifier (a token which can start with _ or samll [a-z] and then there can be the repetition of [a-z] [A-Z] [0-9] and _) .
following is the SLR-1 parsing table for above SDT.



LR table

State	ACTION						GOTO			
	+	*	(	)	id	$	E'	E	T	F
0			s4		s5			1	2	3
1	s6					acc				
2	r_2	s7		r_2		r_2				
3	r_4	r_4		r_4		r_4				
4			s4		s5			8	2	3
5	r_6	r_6		r_6		r_6				
6			s4		s5				9	3
7			s4		s5					10
8	s6			s11						
9	r_1	s7		r_1		r_1				
10	r_3	r_3		r_3		r_3				
11	r_5	r_5		r_5		r_5				

You are required to implement following tasks.

1. Hard code this table in source program. (no need to read from file).

2. Hard code NFA for identifier.

3. Hard code input in the program.

4. Identify all type of token mentioned at start of program and identifier token too.

5. Print all identified tokens and only save tokens which are used in SDT.

6. Parse SDT using SLR-1 grammar.