Assignment 3.

Implement Calculator using Postfix SDT (given below) along with LR parsing.

L	$\rightarrow$	E **n**	{ print($E.val$); }
E	$\rightarrow$	$E_1 + T$	{ $E.val = E_1.val + T.val$; }
E	$\rightarrow$	T	{ $E.val = T.val$; }
T	$\rightarrow$	$T_1 * F$	{ $T.val = T_1.val \times F.val$; }
T	$\rightarrow$	F	{ $T.val = F.val$; }
F	$\rightarrow$	(E)	{ $F.val = E.val$; }
F	$\rightarrow$	**num**	{ $F.val =$ **num**$.lexval$; }

Postfix SDT implementing the desk calculator

For the given Postfix SDT for the arithmetic expressions do the following.

1. In lex using regular definitions define number which is the token **num**.
2. Modify the SDT to include the operator ^ which to the power-of operator
3. Also include the unary minus operator. You should take care of precedence and associativity of these operators. Unary minus will have the next highest precedence to that of the parenthesis and will have right-to-left associativity. Next precedence is for the ^ operator which will have left-to-right associativity. Clearly write your SDT with the added operators.
4. Build LR(1) automaton. (You can do this manually, but report about this clearly by drawing the automaton and explaining the steps)
5. Create CLR parse table for the given grammar.
6. Implement the SDT along with parsing the given input string and produce the output. {Annotate/document your program properly so that it is easy to understand

You should use Lex/flex to define **num**. The rest you can write in C language.

Deadline: 8th Nov Sunday 5pm. This will not be extended.

This is an individual assignment. Plagiarism check will be done. Viva may be there.