Compiler Qns

Sl No	Experiments	Compile and Run
1	Implement a lexical analyzer for a given language using C and the lexical analyzer should ignore redundant spaces, tabs and newlines.	student@localhost$cc lex.c student@localhost$ ./a.out
2	Implementation of Lexical Analyzer using Lex Tool to replace space,tab and newline	student@localhost$lex replace.l student@localhost$gcc -o replace lex.yy.c student@localhost$ ./replace
3	Implementation of Lexical Analyzer using Lex Tool for the identification of positive and negative integers	student@145 $ lex posneg.l student@145 $ cc lex.yy.c student@145 $ ./a.out
4	Implementation of Lexical Analyzer using Lex Tool to count characters ,words and lines in input text	student@localhost $ lex cc.l student@localhost $ cc lex.yy.c student@localhost $ ./a.out
5	Generate a YACC specification to recognize a valid arithmetic expression that use operators +, – , *,/ and parenthesis.	student@localhost $ lex validexpr.l student@localhost $ yacc -d validexpr.y student@localhost$ gcc -o validexpr y.tab.c student@localhost$ ./validexpr
6	Generate a YACC specification to recognize a valid identifier which starts with a letter followed by any number of letters or digits.	student@145$ lex validvar.l student@145$ yacc -d validvar.y student@145 $ cc -o validvar y.tab.c student@145 $ ./validvar
7	Write a program to find First of any given grammar.	cc follow.c ./a.out
8	Generate yacc specification to implement a calculator	student@localhost $ lex calculator.l student@localhost $ yacc -d calculator.y student@localhost $ gcc -o calculator y.tab.c student@localhost $ ./calculator
9	Write a program to find Follow of any given grammar.	cc follow.c ./a.out
10	Implement Intermediate code generation for simple expressions.	cc code.c ./a.out
11	To accept the following language L={0^n1^m, n>=1,m>=0}	[student@localhost]$ lex arithexpr.l [student@localhost]$ cc lex.yy.c [student@localhost]$ ./a.out