Course Code	Course Name	Load Distribution (L T P C)
PCS 601	Compiler Lab	0 0 4 2

Learning Outcomes:

After completing this course students will be able to:

1. Understand programming in LEX and YACC.
2. Tokenize source code using LEX.
3. Design Regular Expression for meaningful words like keyword, operator, separators etc.
4. Implement DFA using LEX Code.
5. Design small working Compiler using LEX and YACC.

Detailed Syllabus:

NOTE: Design LEX/YACC Code for following Set of Program. (Study of LEX/YACC with file-handling is required)

LEX code using Regular Grammar (without file-handling):

1. Design a LEX Code to count the number of lines, space, tab-meta character and rest of characters in a given Input pattern.
2. Design a LEX Code to identify and print valid Identifier of C/C++ in given Input pattern.
3. Design a LEX Code to identify and print integer and float value in given Input pattern.
4. Design a LEX Code for Tokenizing (Identify and print OPERATORS, SEPERATORS, KEYWORDS, IDENTIFERS) the following C-fragment:

```
int p=1,d=0,r=4,
float m=0.0, n=200.0,
while (p <= 3)
{ if(d==0)
{ m= m+n*r+4.5, d++,  }
else
{ r++, m=m+r+1000.0,  }
     p++,  }
```

LEX code using Regular Grammar (with file-handling):

5. Design a LEX Code to count and print the number of total characters, words, white spaces in given 'Input.txt' file.
6. Design a LEX Code to replace white spaces of 'Input.txt' file by a single blank character into 'Output.txt' file.
7. Design a LEX Code to remove the comments from any C-Program given at run-time and store into 'out.c' file.
8. Design a LEX Code to extract all html tags in the given HTML file at run time and store into Text file given at run time.

LEX code using DFA:

9. Design a DFA in LEX Code which accepts string containing even number of 'a' and even number of 'b' over input alphabet {a, b}.
10. Design a DFA in LEX Code which accepts string containing third last element 'a' over input alphabet {a, b}.
11. Design a DFA in LEX Code to Identify and print Integer & Float Constants and Identifier.

YACC/LEX code:

12. Design YACC/LEX code to recognize valid arithmetic expression with operators +, -, * and /.
13. Design YACC/LEX code to evaluate arithmetic expression involving operators +, -, * and / without operator precedence grammar & with operator precedence grammar.
14. Design YACC/LEX code that translates infix expression to postfix expression.
15. Design Desk Calculator using YACC/LEX code.