Index

<table>
<tr><th>No.</th><th>Practical Statement</th><th>Date</th><th>Page No.</th><th>Sign</th></tr>
<tr><td>1</td><td>Develop a 'C' Program finds white spaces, number of newline characters from the given input.</td><td></td><td></td><td></td></tr>
<tr><td>2</td><td>Implement a lexical analyzer (Scanner program) to recognize identifiers, keywords and constants from the given input file and store them separately.</td><td></td><td></td><td></td></tr>
<tr><td>3</td><td>Write a C program to simulate lexical analyzer to validate arithmetic operators, relational operators and logical operators.</td><td></td><td></td><td></td></tr>
<tr><td>4</td><td>Convert the following regular expression (R.E.) into DFA and Write a‘C’ program to simulate the DFA for given input Strings.
(i) a(a|b)* ab. (ii) digit(digit)*(.digit(digit)*|ε)</td><td></td><td></td><td></td></tr>
<tr><td>5</td><td>Implement Recursive Descent Parser program in ‘C’ for the following Grammar.
P ---> E '#'
E ---> T {('+'|'-') T}
T ---> S {('*'|'/') S}
S ---> F '^' S | F
F ---> D | '(' E ')'
D ---> 0|1|......|9.
Write a program in a way that it will trace the processing of different non-terminals of above grammar for given input string.</td><td></td><td></td><td></td></tr>
<tr><td>6</td><td>Write a program to remove left recursion from a given grammar.</td><td></td><td></td><td></td></tr>
<tr><td>7</td><td>Write a program to left factor the given grammar.</td><td></td><td></td><td></td></tr>
<tr><td>8</td><td>Finding “First” set
Input: The string consists of grammar symbols.
Output: The First set for a given string.
Explanation:
The student has to assume a typical grammar. The program when run will ask for the string to be entered. The program will find the First set of the given string.</td><td></td><td></td><td></td></tr>
<tr><td>9</td><td>Write a program to illustrate steps of LL (1) parser for the given parsing table.</td><td></td><td></td><td></td></tr>
<tr><td>10</td><td>Implement a C program to implement operator precedence parsing.</td><td></td><td></td><td></td></tr>
<tr><td>11</td><td>Generate 3-tuple intermediate code for given infix expression.</td><td></td><td></td><td></td></tr>
<tr><td>12</td><td>Study the following compiler construction tools:
a) LEX b) YACC</td><td></td><td></td><td></td></tr>
</table>