Set-10

1. In a class of Grade 3, Mathematics Teacher asked for the Acronym PEMDAS?. All of them are thinking for a while. A smart kid of the class Kishore of the class says it is Parentheses, Exponentiation, Multiplication, Division, Addition, Subtraction. Can you write a C Program to help the students to understand about the operator precedence parsing for an expression containing more than one operator, the order of evaluation depends on the order of operations.

   Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

// Function to return precedence of operators
int precedence(char op) {
  if (op == '+' || op == '-') return 1;
  if (op == '*' || op == '/') return 2;
  if (op == '^') return 3;
  return 0;
}

// Function to perform arithmetic operations
int applyOperation(int a, int b, char op) {
  switch (op) {
    case '+': return a + b;
    case '-': return a - b;
    case '*': return a * b;
    case '/': return a / b;
    case '^': {
      int result = 1;
      for (int i = 0; i < b; i++) result *= a;
      return result;
    }
  }
  return 0;
}

// Function to evaluate an expression
int evaluate(char *expression) {
  int values[100], valuesTop = -1;
  char operators[100];
  int operatorsTop = -1;

  for (int i = 0; i < strlen(expression); i++) {
    if (expression[i] == ' ') continue;

    if (isdigit(expression[i])) {
      int val = 0;
      while (i < strlen(expression) && isdigit(expression[i])) {
        val = (val * 10) + (expression[i] - '0');
        i++;
      }
      values[++valuesTop] = val;
      i--;
    } else if (expression[i] == '(') {
      operators[++operatorsTop] = expression[i];
    } else if (expression[i] == ')') {
      while (operatorsTop != -1 && operators[operatorsTop] != '(') {
        int val2 = values[valuesTop--];
        int val1 = values[valuesTop--];
        char op = operators[operatorsTop--];
        values[++valuesTop] = applyOperation(val1, val2, op);
      }
      operatorsTop--; // Pop '('
    } else {
      while (operatorsTop != -1 && precedence(operators[operatorsTop]) >=
precedence(expression[i])) {
        int val2 = values[valuesTop--];
        int val1 = values[valuesTop--];
        char op = operators[operatorsTop--];
        values[++valuesTop] = applyOperation(val1, val2, op);
      }
      operators[++operatorsTop] = expression[i];
    }
  }

  while (operatorsTop != -1) {
    int val2 = values[valuesTop--];
    int val1 = values[valuesTop--];
    char op = operators[operatorsTop--];
    values[++valuesTop] = applyOperation(val1, val2, op);
  }
  return values[valuesTop];
}

int main() {
  char expression[100];
  printf("Enter an arithmetic expression: ");
  fgets(expression, sizeof(expression), stdin);
  expression[strcspn(expression, "\n")] = 0; // Remove newline character

  printf("Result: %d\n", evaluate(expression));
  return 0;
}
```

2. Write a LEX program which adds line numbers to the given C program file and display the same in the standard output.

Input Source Program: (sample.c)

```
#define PI 3.14
#include<stdio.h>
#include<conio.h>
 void main()
{
int a,b,c = 30;
printf("hello");
}
```

Code:

```
%{
#include <stdio.h>
int line_number = 1;
%}

%%
.* \n { printf("%d %s", line_number++, yytext); }

%%

int main() {
  yylex();
  return 0;
}
```

3.Write a LEX Program to convert the substring abc to ABC from the given input string.

Code:

```
%{
#include <stdio.h>
%}

%%
abc { printf("ABC"); }
. { printf("%s", yytext); }

%%

int main() {
   yylex();
   return 0;
}
```

4. A networking company wants to validate the URL for their clients. Write a LEX program to implement the same.

Code:

```
%{
#include <stdio.h>
#include <regex.h>
%}

%%
(http|https)://[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}(/[a-zA-Z0-9@:%_+.~#?&/=]*)? {
   printf("Valid URL: %s\n", yytext);
}
.* {
   printf("Invalid URL: %s\n", yytext);
}

%%

int main() {
   printf("Enter a URL to validate: \n");
   yylex();
   return 0;
}
```