```
1.)check wheather identifier or not:
%{
%}
%%

[a-zA-Z][a-zA-Z0-9]+ {printf("\frac{1}{2}\) is identifier",yytext);}
.+ {printf("\frac{1}{2}\) is Keyword",yytext);}
%%
int yywrap(){}
int main()
{
printf("Enter the input:");
yylex();
}
output:
```

```
Command Prompt - a.exe
```

```
Microsoft Windows [Version 10.0.19045.4412]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Owner>cd downloads
C:\Users\Owner\Downloads>cd lex code1
C:\Users\Owner\Downloads\lex code1>set path=C:\Program Files\GnuWin32\bin
C:\Users\Owner\Downloads\lex code1>flex identifier.l
C:\Users\Owner\Downloads\lex code1>set path=C:\MinGW\bin
C:\Users\Owner\Downloads\lex code1>gcc lex.yy.c
C:\Users\Owner\Downloads\lex code1>a.exe
Enter the input:tharun
is identifier
 is identifier
if and else
if and else is Keyword
```

2.) 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.

```
#include<stdio.h>
#include<conio.h>
int main()
{
      char s[5];
      printf("\forall n Enter any operator:");
```

```
gets(s);
switch(s[0])
{
      case'>':
             if(s[1]=='=')
                    printf("\forall n Greater than or equal");
              else
                    printf("\forall n Greater than");
              break;
       case'<':
             if(s[1]=='=')
                    printf("\forall n Less than or equal");
              else
                    printf("\forall n Less than");
              break;
       case'=':
             if(s[1]=='=')
                    printf("\u00e4nEqual to");
              else
                    printf("\forall n Assignment");
              break;
       case'!':
             if(s[1]=='=')
                    printf("\u00e4nNot Equal");
              else
```

```
printf("\forall n Bit Not");
       break;
case'&':
       if(s[1]=='&')
              printf("\frac{1}{2}nLogical AND");
       else
              printf("\forall n Bitwise AND");
       break;
case'|':
       if(s[1]=='|')
              printf("\u00e4nLogical OR");
       else
              printf("\forall nBitwise OR");
       break;
case'+':
       printf("\forall n Addition");
       break;
case'-':
       printf("\u00e4nSubstraction");
       break;
case'*':
       printf("\frac{1}{2}nMultiplication");
       break;
case'/':
       printf("\frac{1}{2} n Division");
```

```
break;
             case'%':
                   printf("Modulus");
                   break;
             default:
                   printf("\forall n Not a operator");
      }
}
 C:\Users\Owner\Documents\recognizze the operators.exe
 Enter any operator:<=
 Less than or equal
Process exited after 3.753 seconds with return value 0
Press any key to continue . . .
3.)#include <stdio.h>
#include <ctype.h>
int main() {
    char ch;
    int charCount = 0, wordCount = 0, lineCount = 0;
    int inWord = 0;
    printf("Enter text (Ctrl+D to end):\forall n");
    while ((ch = getchar()) != EOF) {
```

```
charCount++;
    if (ch == '\frac{1}{2}n') {
         lineCount++;
    }
    if (isspace(ch)) {
        inWord = 0;
    } else if (!inWord) {
         inWord = 1;
         wordCount++;
    }
}
// To account for the last line if it doesn't end with a newline
if (charCount > 0 \&\& ch != '\frac{1}{2}n')  {
    lineCount++;
}
printf("Characters: %d\u00e4n", charCount);
printf("Words: %d\u00e4n", wordCount);
printf("Lines: %d\forall n", lineCount);
return 0;
```

}

```
C:\Users\Owner\Documents\lines char word.exe
Enter text (Ctrl+D to end):
 went to my village to meet my relatives
 went to my native place to meet my parents
Characters: 87
Words: 19
Lines: 3
Process exited after 5.931 seconds with return value 0
Press any key to continue . . .
4.)#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
int tempVarCount = 0; // Counter for temporary variables
// Function to generate a new temporary variable
char* newTemp() {
    static char temp[5];
    sprintf(temp, "t%d", tempVarCount++);
    return temp;
}
// Function to print three-address code
void generateTAC(char* left, char op, char* right, char* result) {
    printf("%s = %s %c %s\fmathbf{s}\text{\text{\text{w}}}", result, left, op, right);
```

```
// Recursive function to parse the expression and generate TAC
char* parseExpression(char* expr, int start, int end) {
    int i, lastOp = -1, opPosition = -1, parentheses = 0;
    // Find the last operator in the expression that is outside of any
parentheses
    for (i = start; i \le end; i++) {
        if (expr[i] == '(') {
            parentheses++;
        } else if (expr[i] == ')') {
            parentheses--;
        } else if (parentheses == 0 \&\& (expr[i] == '+' || expr[i] == '-')) {
            lastOp = i;
        } else if (parentheses == 0 \&\& (expr[i] == '*' || expr[i] == '/') \&\&
lastOp == -1) {
            opPosition = i;
        }
    }
    if (lastOp == -1) {
        lastOp = opPosition;
    }
```

}

```
if (lastOp == -1) {
        if (expr[start] == '(' && expr[end] == ')') {
            return parseExpression(expr, start + 1, end - 1);
        } else {
            char* operand = (char*)malloc(2);
            operand[0] = expr[start];
            operand[1] = '\(\frac{1}{4}0'\);
            return operand;
        }
    }
    char* left = parseExpression(expr, start, lastOp - 1);
    char* right = parseExpression(expr, lastOp + 1, end);
    char op = expr[lastOp];
    char* result = newTemp();
    generateTAC(left, op, right, result);
    return result;
int main() {
    char expr[100];
    printf("Enter an arithmetic expression: ");
```

}

```
int len = strlen(expr);

parseExpression(expr, 0, len - 1);

return 0;
}

C:\Users\Owner\Documents\three adress code.exe

Enter an arithmetic expression: (a+b)*c
t0 = a + b
t1 = t1 * c

Process exited after 31.61 seconds with return value 0
Press any key to continue . . .
```