PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
int isKeyword(char buffer[]){
char keywords[32][10] =
{"auto","break","case","char","const","continue","default",
"do","double","else","enum","extern","float","for","goto",
"if","int","long","register","return","short","signed",
"sizeof","static","struct","switch","typedef","union",
"unsigned","void","volatile","while"};
int i, flag = 0;
for(i = 0; i < 32; ++i){
if(strcmp(keywords[i], buffer) == 0){
flag = 1;
break;
}}
return flag;
}
int main(){
char ch, buffer[15], operators[] = "+-*/%=";
FILE *fp;
int i,j=0;
fp = fopen("program.txt","r");
if(fp == NULL){
printf("error while opening the file\n");
exit(0);
}
while((ch = fgetc(fp)) != EOF){
for(i = 0; i < 6; ++i){
if(ch == operators[i])
printf("%c is operator\n", ch);
}
if(isalnum(ch)){
```

```
buffer[j++] = ch;
}
else if((ch == ' ' || ch == '\n') && (j != 0)){
buffer[j] = '\0';
j = 0;
if(isKeyword(buffer) == 1)
printf("%s is keyword\n", buffer);
else
printf("%s is indentifier\n", buffer);
}
}
fclose(fp);
return 0;
}
```

Program.txt

```
Void main()
{
Int a,b,c;
c=a+b;
}
```

OUTPUT

```
void is keyword
main is identifier
int is keyword
a is identifier
b is identifier
c is identifier
c is identifier
= is operator
a is identifier
+ is operator
b is identifier
```

PROGRAM

```
//Implementation of Lexical Analyzer using Lex tool
%{
int COMMENT=0;
%}
identifier [a-zA-Z][a-zA-Z0-9]*
%%
#.* {printf("\n%s is a preprocessor directive",yytext);}
int |float |char |double |while |for |struct |typedef |do |if |break |continue |void
|switch |return |else |goto
printf("\n\t%s is a keyword",yytext);}
"/*" {COMMENT=1;}{printf("\n\t %s is a COMMENT",yytext);}
{identifier}\( {if(!COMMENT)printf("\nFUNCTION \n\t%s",yytext);}
\{ {if(!COMMENT)printf("\n BLOCK BEGINS");}
\} {if(!COMMENT)printf("BLOCK ENDS ");}
{identifier}(\[[0-9]*\])? {if(!COMMENT) printf("\n %s IDENTIFIER",yytext);}
\".*\" {if(!COMMENT)printf("\n\t %s is a STRING",yytext);}
[0-9]+ {if(!COMMENT) printf("\n %s is a NUMBER ",yytext);}
\)(\:)? {if(!COMMENT)printf("\n\t");ECHO;printf("\n");}
\( ECHO;
= {if(!COMMENT)printf("\n\t %s is an ASSIGNMENT OPERATOR",yytext);}
\<= |\>= |\< |== |\> {if(!COMMENT) printf("\n\t%s is a RELATIONAL
OPERATOR",yytext);}
%%
int main(int argc, char **argv)
{
FILE *file;
file=fopen("var.c","r");
if(!file)
{
printf("could not open the file");
exit(0);
}
yyin=file;
yylex();
```

```
printf("\n");
return(0);
}
int yywrap()
{
return(1);
}
```

INPUT

```
//var.c
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c;
a=1;
b=2;
c=a+b;
printf("Sum:%d",c);
}
```

OUTPUT

#include<stdio.h> is a preprocessor directive
#include<conto.h> is a preprocessor directive
void is a keyword
FUNCTION
main(
)
BLOCK BEGINS
int is a keyword
a IDENTIFIER,
b IDENTIFIER,
C IDENTIFIER;
a IDENTIFIER
= is an ASSIGNMENT OPERATOR
1 is a NUMBER;
b IDENTIFIER
= is an ASSIGNMENT OPERATOR
2 is a NUMBER ;
C IDENTIFIER
= ts an ASSIGNMENT OPERATOR
a IDENTIFIER+ b IDENTIFIER;
FUNCTION
printf(
"Sum: %d" is a STRING,
C IDENTIFIER
)
BLOCK ENDS

PROGRAM

```
%{
        #include<stdio.h>
        int lines = 0, words = 0, letters = 0, num = 0, spl_char = 0, total = 0;
%}
%%
\n { lines++; words++;}
[\t ' '] words++;
[a-zA-Z] letters++;
[0-9] num++;
. spl_char++;
%%

int main(void)
{
   yylex();
        total = letters + num + spl_char;
        printf("\n%d lines.", lines);
        printf("\n%d words.", words);
        printf("\n%d characters.\n",total);
}

int yywrap()
{
        return 1;
}
```

OUTPUT

```
hello
How are you?
I am fine
Ok bye
```

4 lines.
9 words.
27 characters

PROGRAM

```
%{
#include<stdio.h>
%}
%%
[ \n\t] {printf("");}
.       {printf("%s",yytext);}
%%

int main()
{
        FILE *fp;
        fp = fopen("input.c", "r");
        yyin = fp;
        yylex();
        return 0;
}
int yywrap() {
   return 1;
}
```

Input.c

```
#include<stdio.h>
void main(){
   int a    =    10;
   int b     =   20;
   int x  = a+b;
   if(x>10){
      printf("x is larger than 10");
   }
}
```

OUTPUT

```
#include<stdio.h>voidmain(){inta=10;intb=20;intx=a+b;if(x>10){printf("xislar
gerthan10");}}
```

PROGRAM

```
%{
        #include <stdio.h>
        int vowels = 0;
        int consonants = 0;
%}

%%
[aeiouAEIOU] {vowels++;}
[\t ] {ECHO;}
. {consonants++;}
%%

int main()
{
        yylex();
        printf("Vowels: %d\n", vowels);
        printf("Consonants: %d\n", consonants);
        return 0;
}

int yywrap()
{
        return 1;
}
```

OUTPUT

hello how are you ? I am fine. Okay Bye!

Vowels: 14
Consonants: 17

PROGRAM

valid_id.l

```
%{
   #include "valid_id.tab.h"
%}

%%
[a-zA-Z_][a-zA-Z_0-9]* return letter;
[0-9]                  return digit;
.                     return yytext[0];
\n                     return 0;
%%

int yywrap()
{
        return 1;
}
```

valid_id.y

```
%{
   #include <stdio.h>
   int valid = 1;
%}
%token digit letter
%%
start: letter s
s: letter s | digit s |;
%%
int yyerror()
{
   printf("Invalid identifier.\n");
   valid = 0;
   return 0;
}
```

```
int main()
{
    printf("Enter identifier: ");
    yyparse();
    if(valid)
    {
        printf("Valid identifier.\n");
    }
}
```

OUTPUT

```
Enter identifier: hello
Valid identifier.

Enter identifier: 1erwq
Invalid identifier.

Enter identifier: @qwrty123
Invalid identifier.

Enter identifier: compiler@123
Invalid identifier.

Enter identifier: compiler123
Valid identifier.
```

PROGRAM

valid_ar.l

```
%{
        #include "valid_ar.tab.h"
%}
%%
[a-zA-Z_][a-zA-Z_0-9]* return id;
[0-9]+(\.[0-9]*)?       return num;
[+/*]                    return op;
.                       return yytext[0];
\n                       return 0;
%%
int yywrap()
{
        return 1;
}
```

valid_ar.y

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

%token num id op

%%
start: id '=' s ';'
s: id x | num x | '-' num x | '(' s ')' x;
x: op s | '-' s |;
%%

int yyerror()
{
   valid = 0;
```

```
    printf("Invalid expression.\n");
    return 0;
}

int main()
{
    printf("Enter the expression: ");
    yyparse();

    if(valid)
    {
        printf("Valid expression.\n");
    }
}
```

OUTPUT

```
Enter the expression: a+b=c;
Invalid expression.

Enter the expression: a=b+c;
Valid expression.

Enter the expression: a+b=c+d;
Invalid expression.

Enter the expression: a=b*c+d/e;
Valid expression.
```

PROGRAM

calc.l

```
%{
        #include "calc.tab.h"
        extern int yylval;
%}
%%
[0-9]+ {
        yylval=atoi(yytext);
        return NUMBER;
}
[ \t]    ;
.       return yytext[0];
%%
int yywrap()
{
        return 1;
}
```

calc.y

```
%{
   #include <stdio.h>
   int flag = 0;
%}

%token NUMBER
%left '+' '-'
%left '*' '/' '%'
%left '(' ')'

%%
S: E{
        printf("Result: %d\n",$$);
        return 0;
```

```
};

E:
        E'+'E {$$=$1+$3;} |
        E'-'E {$$=$1-$3;} |
        E'*'E {$$=$1*$3;} |
        E'/'E {$$=$1/$3;} |
        E'%'E {$$=$1%$3;} |
        '('E')' {$$=$2;} |
        NUMBER {$$=$1;};
%%

void main()
{
        printf("Enter expression: ");
        yyparse();
        if (flag == 0)
                printf("Valid expression.\n");
}

void yyerror()
{
        printf("Invalid expression.\n");
        flag = 1;
}
```

OUTPUT

```
Enter expression: (5+6)*(3+2)/3;
Result: 18
Valid expression.

Enter expression: (1+2*3)/(1+8);
Result: 0
Valid expression.
```

PROGRAM

```
%{
#include<stdio.h>
#include<string.h>
int i;
%}
%%
[a-zA-Z]* {
   for(i = 0; i <= yyleng - 3; i++) {
      if((yytext[i] == 'a') && (yytext[i+1] == 'b') && (yytext[i+2] == 'c')) {
         yytext[i] = 'A';
         yytext[i+1] = 'B';
         yytext[i+2] = 'C';
      }
   }
   printf("%s", yytext);
}
[\t]* {/* do nothing */}
.* { ECHO; }
\n { printf("%s", yytext); }
%%
int main() {
   yylex();
   return 0;
}
int yywrap() {
   return 1;
}
```

OUTPUT

cdeabcfgh
cdeABCfgh

abccompiler
ABCcompiler