Assignment No 01

Write a program using LEX specifications to implement lexical analysis phase of compiler to generate tokens of subset of 'C' program

assignmentOne.l

```
%{
#include<stdio.h>
%}
%%
[/][/].* {printf("\n Single line comment1 : %s",yytext);}
"/*"[^*]*"*/" {printf("\n Multi line comment : %s",yytext);}
# {printf("\n Processing Directives : %s",yytext);}
include|printf|int|void|main {printf("\n Keywords : %s",yytext);}
"<"|">"|"("|")"|";"|","|"{"|"}" {printf("\n Punctuation : %s",yytext);}
[a-z]+[.][h] {printf("\n Header files : %s",yytext);}
["].*["] {printf("\n Litrels : %s",yytext);}
[a-zA-Z][a-zA-Z0-9_] {printf("\n Identifier : %s",yytext);}
[0-9]+ {printf("\n Integer Number : %s",yytext);}
[0-9]+(\.[0-9]+) {printf("\n Decimal Number : %s",yytext);}

"+"|"-"|"=" {printf("\n Operators : %s",yytext);}
%%
int yywrap()
{
return 1;
}
int main()
{
yyin=fopen("pro.c","r");
yylex();
return 0;
}
```

pro.c

```
#include<stdio.h>

void fun(){

	printf("Hello this is user defined function");

}

int main(){

	//variable

	int rno=77;

	float marks=8.88;

	printf("Akshada Phopse");

	fun();

	return 0;

	/*hello this

	is multiline comment*/

}
```

Commands to Run Program

1. lex assignmentOne.l
2. gcc lex.yy.c
3. ./a.out

Assignment No 02

Write a LEX program to display word, character and line counts for a sample input text file

assignmentTwo.l

```
%{

#include <stdio.h>

int wc = 0, lc = 0, cc = 0, dc = 0, vc = 0;

%}


%%

[aeiouAEIOU]     { vc++; cc++; wc++;}

[0-9]            { dc++; cc++; }

\n               { lc++; cc++; }
```

```
[ \t]+          { cc += yyleng; }
[^ \t\n]+         { wc++; cc += yyleng; }
%%

int yywrap()
{
    return 1;
}

int main()
{
    yyin = fopen("atwo.txt", "r");
    yylex(); // Perform lexical analysis
    printf("Number of Lines : %d\n", lc);
    printf("Number of Words : %d\n", wc);
    printf("Number of Characters : %d\n", cc);
    printf("Number of Digits : %d\n", dc);
    printf("Number of Vowels : %d\n", vc);
    return 0;
}
```

atwo.txt

```
This is 2 nd assignment a e i o u
```

Commands to Run Program

1. lex assignmentTwo.l
2. gcc lex.yy.c
3. ./a.out

Assignment No 03

Write a program using YACC specifications to implement syntax analysis phase of compiler to validate type and syntax of variable declaration in C program.

assignThree.l

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

%%
"int"    { return INT; }
"float"  { return FLOAT; }
"char"   { return CHAR; }
[a-zA-Z_][a-zA-Z0-9_]*  { return ID; }
","      { return COMMA; }
";"      { return SEMICOLON; }
[ \t\n]  { /* Ignore whitespace */ }
.        { return yytext[0]; }
%%

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

assignThree.y

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

int yyerror(char *str);
int yywrap();
%}

%token INT FLOAT CHAR ID COMMA SEMICOLON

%%

Stmt: Type VarList SEMICOLON { printf("Valid Declaration\n"); }
     | error SEMICOLON       { printf("Invalid Declaration\n"); }
     ;
```

```
Type: INT | FLOAT | CHAR;


VarList: ID

       | ID COMMA VarList;


%%


int yyerror(char *str) {

    printf("Syntax Error: %s\n", str);

    return 0;

}


int main() {

    printf("Enter a variable declaration:\n");

    yyparse();

    return 0;

}
```

Commands to Run Program

1. lex assignThree.l
2. yacc –d assignThree.y
3. gcc lex.yy.c y.tab.c
4. ./a.out

Assignment No 04

Write a program using YACC specifications to implement calulator to perform various arithmetic operations

assignFour.l

```
%{

#include "y.tab.h"

%}


%%

[0-9]           { yylval = atoi(yytext); return N; }
```

```
[ \t ]
"\n"                    { return 0;}
.                  {return yytext[0]; }
%%

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

assignFour.y

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

int yyerror(char *str);
int yywrap();
%}

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

%%

A: E { printf("Result is = %d\n", $$); };

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



%%

int yyerror(char *str) {
    printf("Invalid Expression\n");
    return 0;
}
int main() {
    printf("Enter Arithmetic Expression: ");
    yyparse();
    return 0;
}
```

Commands to Run Program

1. lex assignFour.l
2. yacc –d assignFour.y
3. gcc lex.yy.c y.tab.c
4. ./a.out