                                                                        LEX PROGRAMS

Lex Program 1

%{

#include<stdio.h>

int lines=0, words=0, characters=0, num=0,spaces=0, spl\_char=0;

%}

%%

\n { lines++; words++;}

[' '] {words++; spaces++; }

[A-Za-z0-9] characters++;

. spl\_char++;

%%

main(void)

{

yyin= fopen("myfile.txt","r");

yylex();

printf(" This File contains ...");

printf("\n\t%d lines", lines);

printf("\n\t%d words",words);

printf("\n\t%d characters", characters);

printf("\n\t%d spaces", spaces);

printf("\n\t%d special characters\n",spl\_char);

}

int yywrap()

{

return(1);

}

MYFILE.TXT

This is my 1st lex program!!!

Cheers!! It works!!:)

OUTPUT

lex 1.l

6c@ISELAB2-41:~$ lex.yy.c

6c@ISELAB2-41:~$ gcc lex.yy.c -ll

6c@ISELAB2-41:~$ ./a.out MYFILE.txt

This File contains ...

   2 lines

   9 words

   34 characters

   7 spaces

   9 special characters

-----------------------------------------------------------------

Lex Program 2

%{

#include<stdio.h>

int com=0;

%}

%s COMMENT

%%

"/\*"[.]\*"\*/" {com++;}

"/\*" {BEGIN COMMENT ;}

<COMMENT>"\*/" {BEGIN 0; com++ ;}

<COMMENT>\n {com++ ;}

<COMMENT>. {;}

\/\/.\* {; com++;}

.|\n {fprintf(yyout,"%s",yytext);};

%%

int yywrap()

{

return 1;

}

main(int argc, char \*argv[])

{

if(argc!=3)

{

printf("Usage: <./a.out> <sourcefile> <destn file>\n");

exit(0);

}

yyin=fopen(argv[1],"r");

yyout=fopen(argv[2],"w");

yylex();

printf("No of comment lines=%d\n",com);

}

IN.TXT

#include<stdio.h>

int main()

{

int a,b,c; /\*varible declaration

bhdfhvjbfdh\*/

printf(“enter two numbers”);

scanf(“%d %d”,&a,&b);

c=a+b;//adding two numbers

printf(“sum is %d”,c);

return 0;

}

OUTPUT

lex 2.l

6c@ISELAB2-41:~$ gcc lex.yy.c -ll

6c@ISELAB2-41:~$ ./a.out IN.txt OUT.txt

No of comment lines=3

----------------------------------------------------------------------------------------------------------------

LEX PROGRAM 3

%{

#include<stdio.h>

#include<string.h>

int noprt=0, nopnd=0, valid=1, top=-1, m, l=0, j=0;

char opnd[10][10], oprt[10][10], a[100];

%}

%%

"(" { top++; a[top]='(' ; }

"{" { top++; a[top]='{' ; }

"[" { top++; a[top]='[' ; }

")" { if(a[top]!='(')

{

valid=0; return;

}

else

top--;

}

"}" { if(a[top]!='{')

{

valid=0; return;

}

else

top--;

}

"]" { if(a[top]!='[')

{

valid=0; return;

}

else

top--;

}

"+"|"-"|"\*"|"/" { noprt++;

strcpy(oprt[l], yytext);

l++;

}

[0-9]+|[a-zA-Z][a-zA-Z0-9\_]\* {nopnd++; strcpy(opnd[j],yytext); j++;}

%%

int yywrap()

{

return 1;

}

main()

{

int k;

printf("Enter the expression.. at end press ^d\n");

yylex();

if(valid==1 && (nopnd-noprt)==1)

{

printf("The expression is valid\n");

printf("The operators are\n");

for(k=0;k<l;k++)

    printf("%s\n",oprt[k]);

for(k=0;k<j;k++)

    printf("%s\n",opnd[k]);

}

else

    printf("The expression is invalid\n");

}

OUTPUT

lex  3.l

6c@ISELAB2-41:~$ gcc  lex.yy.c -ll

6c@ISELAB2-41:~$ ./a.out

Enter the expression.. at end press ^d

(a-b+c\*d)

The expression is valid

The operators are

-

+

\*

a

b

c

d

----------------------------------------------------------------------------------------------------------------

LEX PROGRAM 4

%{

#include<stdio.h>

int id=0;

%}

%%

[a-zA-Z][a-zA-Z0-9\_]\* { id++ ; ECHO; printf("\n");}

.+ { ;}

\n { ;}

%%

int yywrap()

{

return 1;

}

main (int argc, char \*argv[])

{

if(argc!=2)

{

printf("Usage: <./a.out> <sourcefile>\n");

exit(0);

}

yyin=fopen(argv[1],"r");

printf("Valid identifires are\n");

yylex();

printf("No of identifiers = %d\n",id);

}

INPUT.TXT

int

float

78f

90gh

a

d

are

default

printf

OUTPUT

6c@ISELAB2-41:~$ lex 4.l

6c@ISELAB2-41:~$ gcc lex.yy.c -ll

6c@ISELAB2-41:~$ ./a.out INPUT.txt

Valid identifires are

int

float

a

d

are

default

printf

No of identifiers = 7

-----------------------------------------------------------------

**YACC PROGRAMS**

Yacc Program 1

-----lex part-----

%{

#include "y.tab.h"

%}

%%

[0-9]+ { return NUMBER; }

[a-zA-Z][a-zA-Z0-9\_]\* { return ID; }

\n { return NL ;}

. { return yytext[0]; }

%%

int yywrap(void){return 1;}

-----end----------

-----yacc-----------

%{

#include<stdio.h>

#include<stdlib.h>

%}

%token NUMBER ID NL

%left '+' '-'

%left '\*' '/'

%%

stmt : exp NL { printf("Valid Expression\n"); exit(0);}

;

exp : exp '+' exp

| exp '-' exp

| exp '\*' exp

| exp '/' exp

| '(' exp ')'

| ID

| NUMBER

;

%%

int yyerror(char \*msg)

{

printf("Invalid Expression\n");

exit(0);

}

main ()

{

printf("Enter the expression\n");

yyparse();

}

---------end---------

O/P

gedit validid.l

gedit validid.y

6c@ISELAB2-41:~$ lex validid.l

6c@ISELAB2-41:~$ yacc validid.y

6c@ISELAB2-41:~$ gcc lex.yy.c y.tab.c -ll -ly

6c@ISELAB2-41:~$ ./a.out

Enter the expression

ab+56

Valid Expression

6c@ISELAB2-41:~$ ./a.out

Enter the expression

a+

Invalid Expression

----------------------------------------------------------------------------------------------------------------

YACC PROGRAM 2

---------lex part----------

%{

#include "y.tab.h"

%}

%%

[a-zA-Z] { return LETTER ;}

[0-9] { return DIGIT ; }

[\n] { return NL ;}

[\_] { return UND; }

. { return yytext[0]; }

%%

int yywrap(void){return 1;}

----------end---------------

---------yacc-----------------

%{

#include<stdio.h>

#include<stdlib.h>

%}

%token DIGIT LETTER NL UND

%%

stmt : variable NL { printf("Valid Identifiers\n"); exit(0);}

;

variable : LETTER alphanumeric

;

alphanumeric: LETTER alphanumeric

| DIGIT alphanumeric

| UND alphanumeric

| LETTER

| DIGIT

| UND

;

%%

int yyerror(char \*msg)

{

printf("Invalid Expression\n");

exit(0);

}

main ()

{

printf("Enter the variable name\n");

yyparse();

}

lex ly2.l

6c@ISELAB2-41:~$ yacc -d ly2.y

6c@ISELAB2-41:~$ gcc y.tab.c lex.yy.c -ly -ll

6c@ISELAB2-41:~$ ./a.out

Enter the variable name

avh

Valid Identifiers

6c@ISELAB2-41:~$ ./a.out

Enter the variable name

6767ghjghj

Invalid Expression

--------------------------------------------------------------------------------------------------------------------------------

YACC PROGRAM 3

----------lex----------

%{

#include "y.tab.h"

extern int yylval;

%}

%%

[0-9]+ { yylval=atoi(yytext); return NUMBER; }

\n { return NL ;}

. { return yytext[0]; }

%%

int yywrap(void){return 1;}

----------end----------

----------yacc---------

%{

#include<stdio.h>

#include<stdlib.h>

%}

%token NUMBER ID NL

%left '+' '-'

%left '\*' '/'

%%

stmt : exp NL { printf("value = %d\n",$1); exit(0);}

;

exp : exp '+' exp { $$=$1+$3; }

| exp '-' exp { $$=$1-$3; }

| exp '\*' exp { $$=$1\*$3; }

| exp '/' exp { if($3==0)

{

printf("Cannot divide by 0");

exit(0);

}

else

$$=$1/$3;

}

| '(' exp ')' { $$=$2; }

| ID { $$=$1; }

| NUMBER { $$=$1; }

;

%%

int yyerror(char \*msg)

{

printf("Invalid Expression\n");

exit(0);

}

main ()

{

printf("Enter the expression\n");

yyparse();

}

----------end-----------

----------------------------------------------------------------------------------------------------------------

YACC PROGRAM 4

----------lex------------

%{

#include "y.tab.h"

%}

%%

a return A;

b return B;

.|\n return yytext[0];

%%

int yywrap(void){return 1;}

--------------------------

----------yacc-----------

%{

#include<stdio.h>

#include<stdlib.h>

int valid=1;

%}

%token A B

%%

str:S'\n' {return 0;}

S:c S d

|

c : c A

|

d : d B

|

;

%%

int yyerror()

{

printf("Invalid String\n");

exit(0);

}

main()

{

  printf("Enter the string:\n");

  yyparse();

  if(valid==1)

  printf("---valid string---\n");

}