Graphic Era HILL UNIVERSITY
Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)

Term work

on

Compiler Design (PCS 601)

2022-23

Submitted to:	**Submitted by:**
Mr. Aniruddha Prabhu	Abhay Rautela
Asst. Professor	University Roll. No.: 2018059
GEHU, D. Dun	Class Roll. No./Section: 02/B

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

Graphic Era HILL UNIVERSITY

Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)

DEPARTMENT OF CSE
STUDENT LAB REPORT SHEET

Name of Student .... **Mob. No** ....

Photograph Passport Size

Address Permanent ....

Father's Name ……............................. **Occupation** .... **Mob. No** ....

Mother's Name .... **Occupation** .... **Mob. No** ....

Section .... **Branch** ....…**Semester** .... **Class Roll No**............. **GradeA B C**

Local Address .... **Email**…………….................................. **Marks 5 3 1**

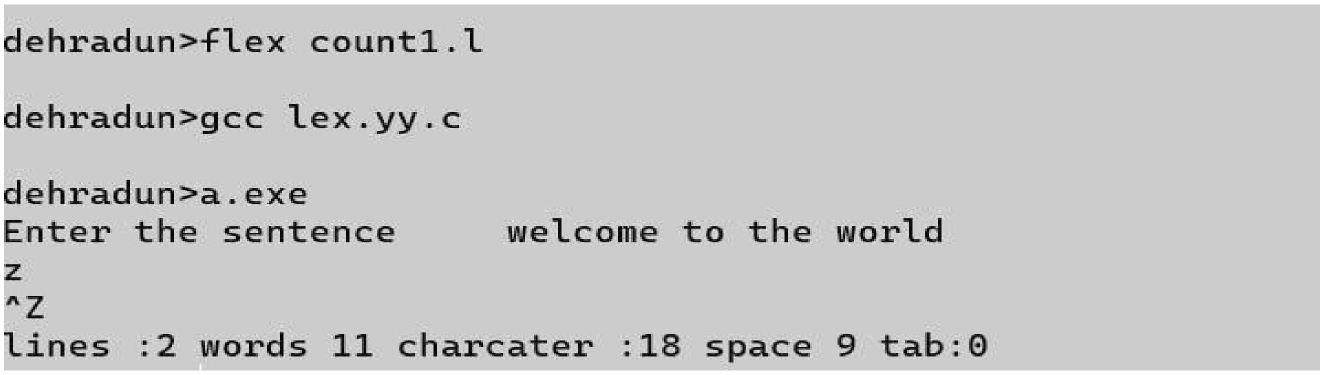
S.No.	Practical	D.O.P.	Date of Submiss ion	Grade (Viva)	Grade (Report File)	Total Marks (out of 10)	Student's Signature	Teacher's Signature
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								

Program 1: Lex code for counting the number of lines, spaces, tabs, the and rest of the characters.

Source Code

```
  %{ int
n,m,t,c;
 %}

 %%

 \n n++;

 \t m++;

 [ ] t++;

 . c++;

 %%

 int yywrap()
 {
   return 1;
 }

 int main()

 { yylex();

 printf("Total number of\nlines=%d \ntabs=%d\nspaces=%d\nchars=%d \n",n,m,t,c); }
```

Output



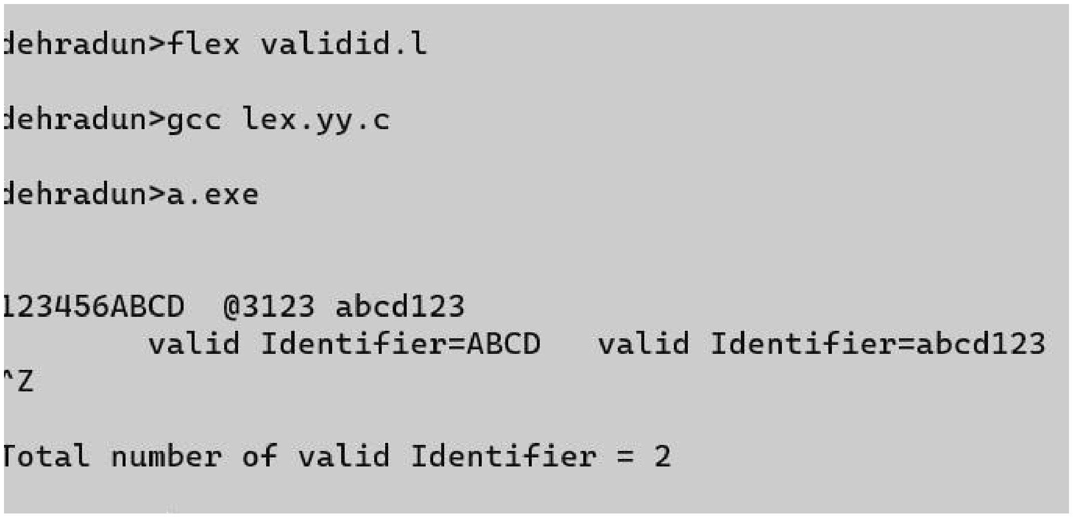
```
dehradun>flex count1.l

dehradun>gcc lex.yy.c

dehradun>a.exe
Enter the sentence     welcome to the world
z
^Z
lines :2 words 11 charcater :18 space 9 tab:0
```

Program 2: Design a LEX Code to identify and print valid Identifier of C/C++ in given Input pattern.

Source Code

```
  %{

   int c=0;

%}

%%

[a-zA-Z_][a-zA-Z0-9_]* {c++; printf("\tvalid Identifier=%s",yytext);}

. ;

%%

int yywrap()
{
  return 1 ;

int main()

{

yylex();

printf("\nTotal number of valid Identifier = %d \n",c);

}
```

Output



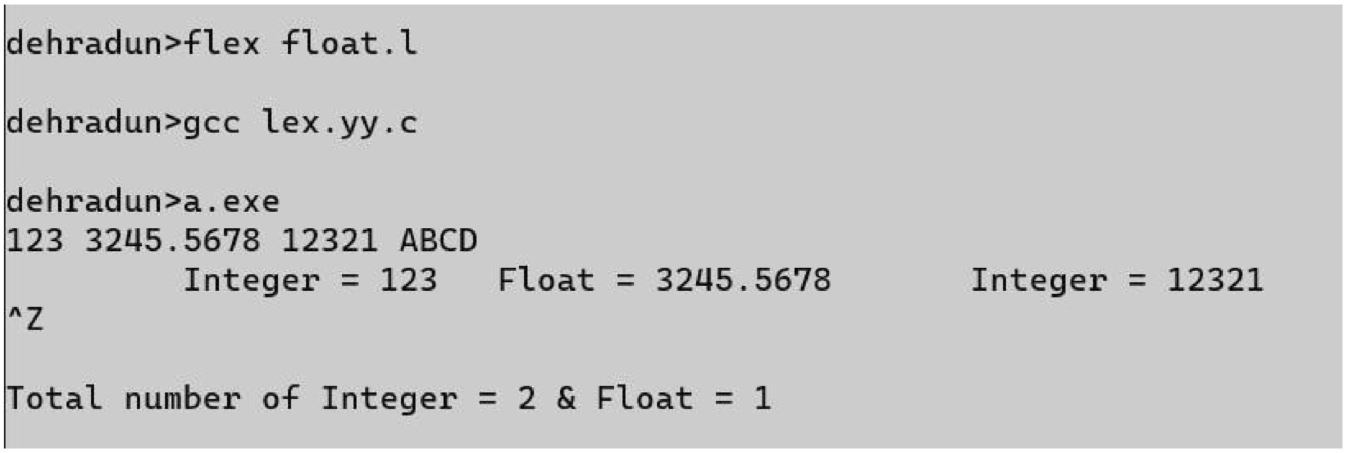
```
dehradun>flex validid.l

dehradun>gcc lex.yy.c

dehradun>a.exe


123456ABCD  @3123 abcd123
        valid Identifier=ABCD   valid Identifier=abcd123
^Z

Total number of valid Identifier = 2
```

Program 3: Design a LEX Code to identify and print integer and float value in given Input pattern.

Source Code

```
%{

  int m=0,n=0;

%}

%%

-?[0-9]+ {m++; printf("\t Integer = %s",yytext);}

-?[0-9]+"."[0-9]+ {n++; printf("\t Float = %s",yytext);}

. ;

%%
int yywrap()
{
 return 1 ;
}

int main()

{

yylex();

printf("\nTotal number of Integer = %d & Float = %d \n",m,n);

}
```

Output



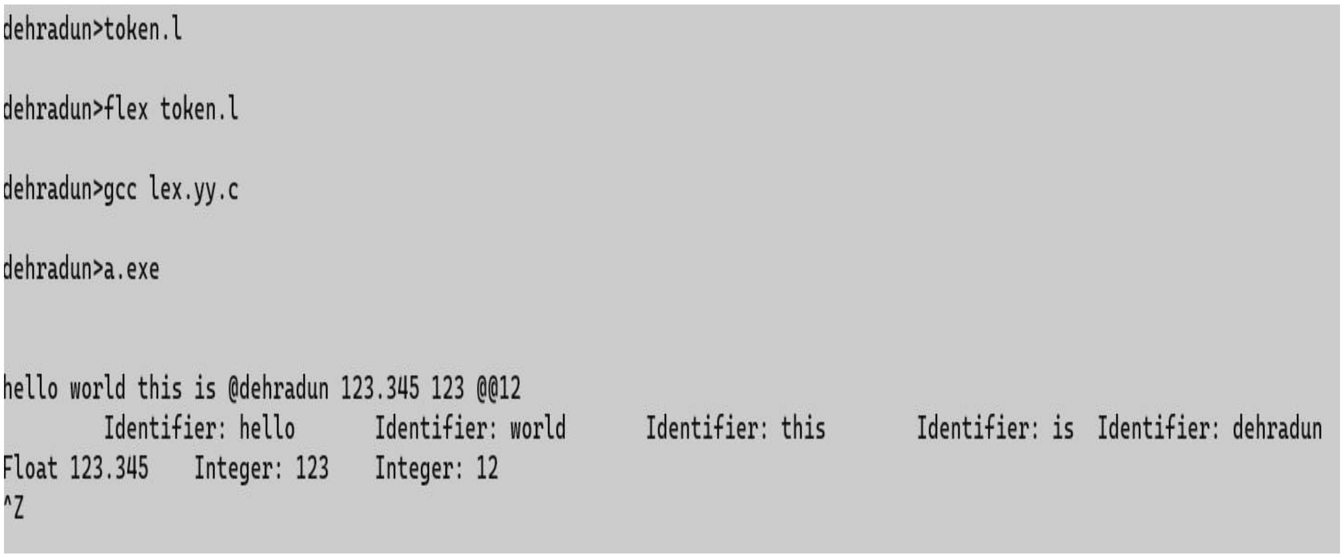
```
dehradun>flex float.l

dehradun>gcc lex.yy.c

dehradun>a.exe
123 3245.5678 12321 ABCD
         Integer = 123   Float = 3245.5678        Integer = 12321
^Z

Total number of Integer = 2 & Float = 1
```

Program 4: Lex code for tokenizing C-code

Source Code

```
 %{

   int n=0;

 %}

 %%

"while"|"if"|"else" {n++; printf("\t keywords: %s",yytext);}

 "int"|"float"       {n++; printf("\t keywords: %s",yytext);}

 [a-zA-Z_][a-zA-Z0-9_]*      {n++; printf("\t Identifier: %s",yytext);} "<="|"=="|"="|"++"|"-

 "|"*"|"+""("|")"|","     {n++; printf("\t operator:

%s",yytext);} "{"|"}"|";"          {n++; printf("\t Seperators: %s",yytext);}

-?[0-9]+"."[0-9]+          {n++; printf("\t Float %s",yytext);}

-?[0-9]+             {n++; printf("\t Integer: %s",yytext);}

.            ;

 %%

 int yywrap()
 {   return 1 ;
 }

 int main()

 { yylex();

 printf("\nTotal number of token = %d \n",n); }

}
```

Output



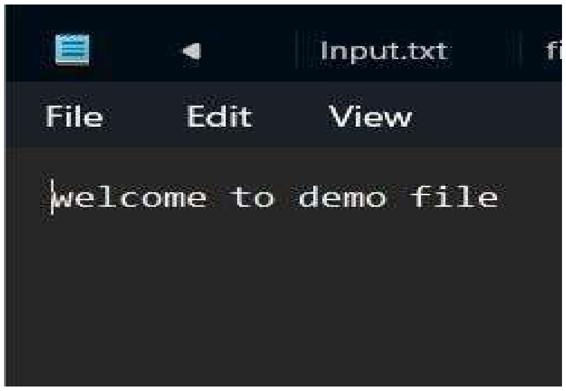
```
dehradun>token.l

dehradun>flex token.l

dehradun>gcc lex.yy.c

dehradun>a.exe


hello world this is @dehradun 123.345 123 @@12
        Identifier: hello       Identifier: world       Identifier: this        Identifier: is  Identifier: dehradun
Float 123.345    Integer: 123    Integer: 12
^Z
```

Program 5: Design a LEX Code to count the number of total characters, words, white spaces in a given “Input.txt” file.

Source Code

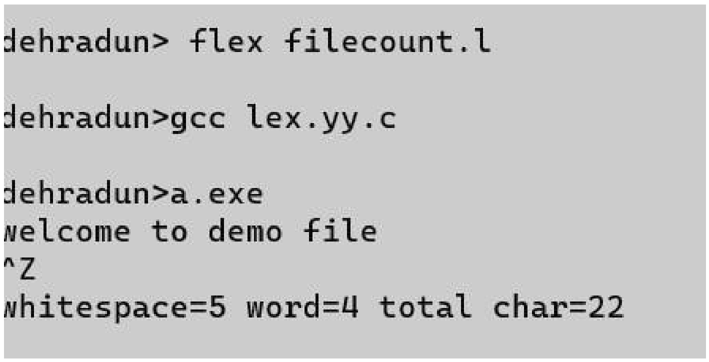
```
%{
    int n,w,c;
%}
%%
  [ \n\t] {n++;}
[^ \n\t]+ {w++;c=c+yyleng;}
%%
int yywrap()
{ return 1
;
} int
main() {
extern FILE *yyin;
yyin=fopen("Input.txt","r"); yylex();
printf("whitespace=%d word=%d total char=%d \n",n,w,n+c);
}
```

Input file



```
welcome to demo file
```

Output



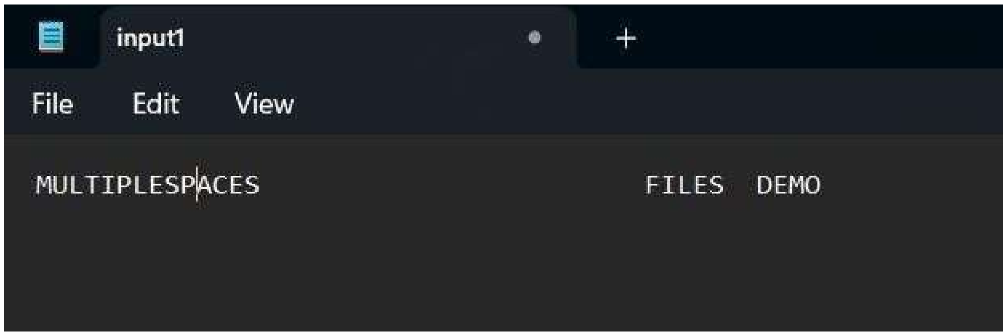
```
dehradun> flex filecount.l

dehradun>gcc lex.yy.c

dehradun>a.exe
welcome to demo file
^Z
whitespace=5 word=4 total char=22
```

Program 6: Lex code for replacing multiple whitespaces by single space

Source Code

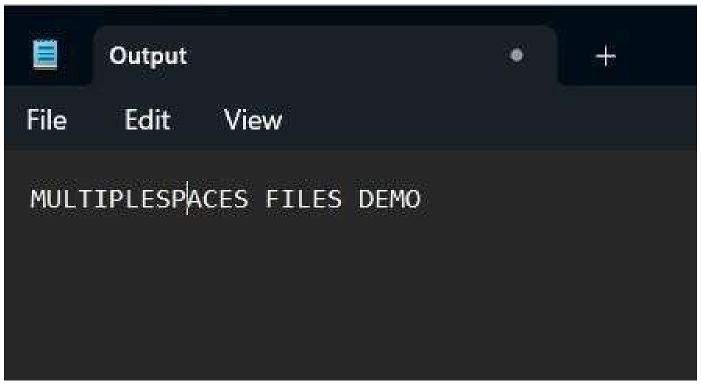
```
 %{
 %}
 %%
.       fprintf(yyout,"%s",yytext);
 %%
 int yywrap()
 { return 1
 ; } int
 main()
 { extern FILE *yyin,*yyout; yyin=fopen("Input.txt","r");
 yyout=fopen("Output.txt","w"); yylex();
 }
```

Source File



```
MULTIPLESPACES                              FILES  DEMO
```

Output File



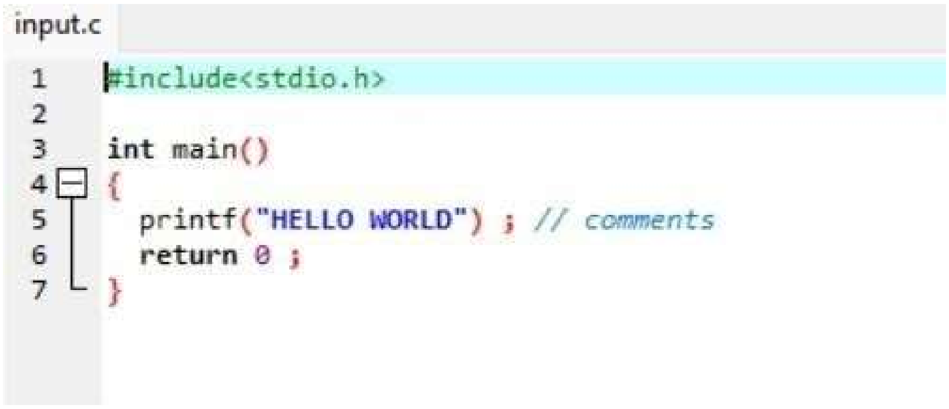
```
MULTIPLESPACES FILES DEMO
```

Program 7: Lex code for removing C-comment from C-program.

Source Code

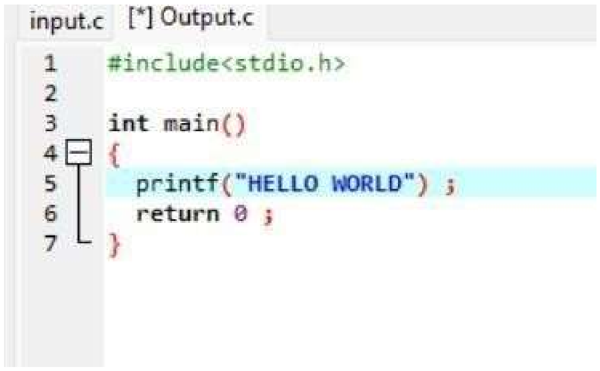
```
%{

%}

%%

"//"[^\n]* ; "/*"([^*]|[*]+[^/])*[*]+"/" ;

. fprintf(yyout,"%s",yytext);

%%

int main()

{ extern FILE *yyin,*yyout;

yyin=fopen("Input.c","r"); yyout=fopen("Out.c","w");

yylex();

}
```

Input File

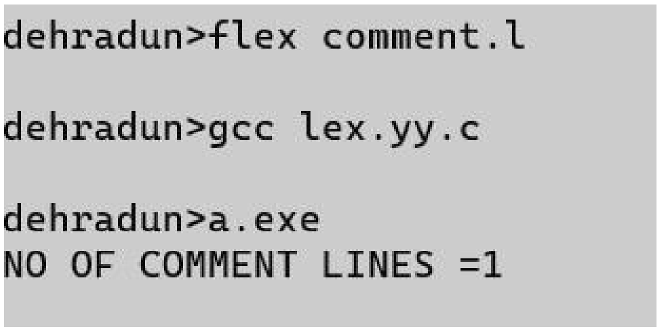


```
#include<stdio.h>

int main()
{
  printf("HELLO WORLD") ; // comments
  return 0 ;
}
```

Output File



```
#include<stdio.h>

int main()
{
  printf("HELLO WORLD") ;
  return 0 ;
}
```



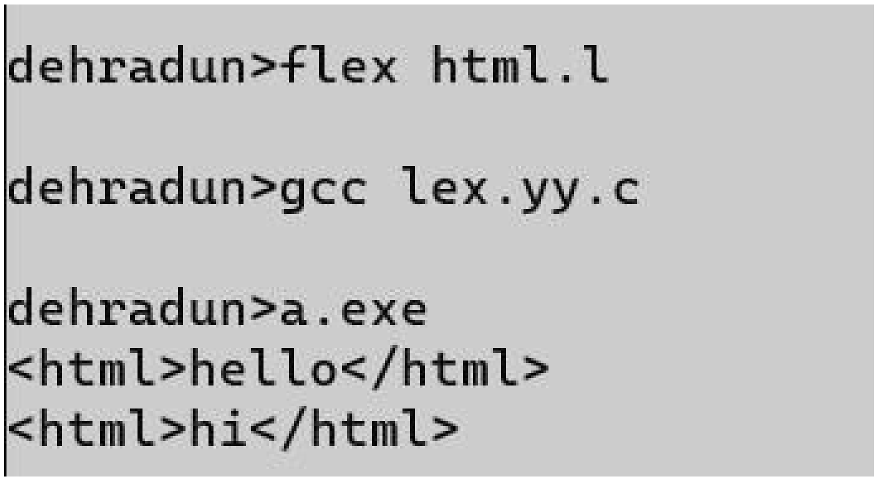
```
dehradun>flex comment.l

dehradun>gcc lex.yy.c

dehradun>a.exe
NO OF COMMENT LINES =1
```

Program 8: Design a LEX Code to extract all html tags in the given HTML file at run time and store into Text file given at run time

Source Code

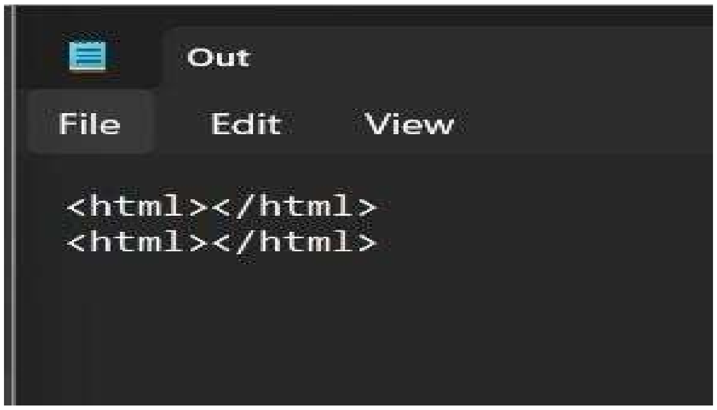
```
%{
%}
%% "<"[^>]*">" ;
. fprintf(yyout,"%s",yytext);
%%
int yywrap()
{ return 1 ; } int main(int
args, char **argv) { extern
FILE *yyin,*yyout;
yyin=fopen(argv[1],"r");
yyout=fopen(argv[2],"w");
yylex();
}
```

Input



```
dehradun>flex html.l

dehradun>gcc lex.yy.c

dehradun>a.exe
<html>hello</html>
<html>hi</html>
```

Output



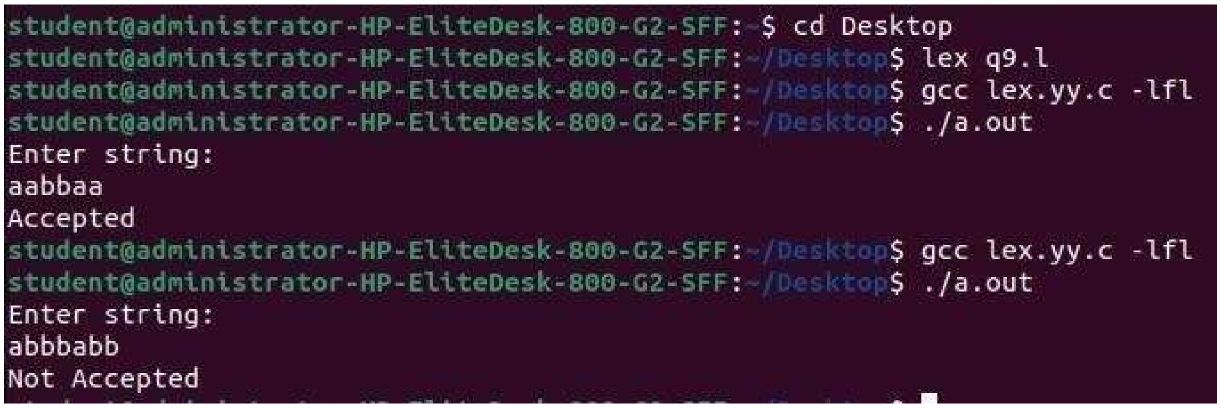
```
<html></html>
<html></html>
```

Program 9: Design a DFA in LEX Code which accepts string containing even number of a’s and even number of b’s over input alphabet {a,b}.

Source Code

```
%{
#include<stdio.h>
%}
%s A B C DEAD
%%
<INITIAL>a BEGIN A; <INITIAL>b
BEGIN B;
<INITIAL>[^ba\n] BEGIN DEAD;
<INITIAL>\n BEGIN INITIAL; {printf("Accepted\n");}
<A>a BEGIN INITIAL;
<A>b BEGIN C;
<A>[^ba\n] BEGIN DEAD;
<A>\n BEGIN INITIAL; {printf("Not Accepted\n");}
<B>a BEGIN C;
<B>b BEGIN INITIAL;
<B>[^ba\n] BEGIN DEAD;
<B>\n BEGIN INITIAL; {printf("Not Accepted\n");}
<C>a BEGIN B;
<C>b BEGIN A;
<C>[^ba\n] BEGIN DEAD;
<C>\n BEGIN INITIAL; {printf("Not Accepted\n");}
DEAD>[^\n] BEGIN DEAD;
<DEAD>\n BEGIN INITIAL; {printf("Invalid\n");}
%%
int yywrap() {return 1;} int
main()          {printf("Enter
String\n");  yylex();  return
0; }
```

Output



```
student@administrator-HP-EliteDesk-800-G2-SFF:~$ cd Desktop
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop$ lex q9.l
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop$ gcc lex.yy.c -lfl
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop$ ./a.out
Enter string:
aabbaa
Accepted
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop$ gcc lex.yy.c -lfl
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop$ ./a.out
Enter string:
abbbabb
Not Accepted
```

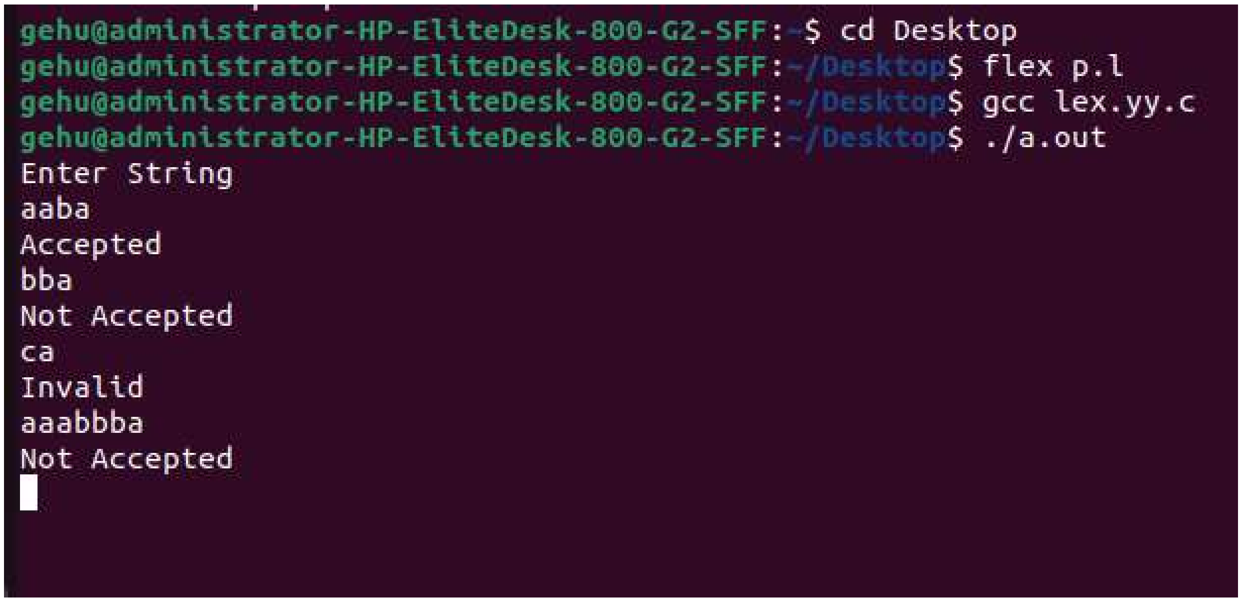
Program 10: Design a DFA in LEX Code which accepts string containing third last element 'a' over input alphabet {a, b}.

Source Code

```
%{
%}
%s A B C D E F G DEAD
%%
<INITIAL>b BEGIN INITIAL;
<INITIAL>a BEGIN A;
<INITIAL>[^ab\n] BEGIN DEAD;
<INITIAL>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<A>b BEGIN F;
<A>a BEGIN B;
<A>[^ab\n] BEGIN DEAD;
<A>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<B>b BEGIN D;
<B>a BEGIN C;
<B>[^ab\n] BEGIN DEAD;
<B>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<C>b BEGIN D;
<C>a BEGIN C;
<C>[^ab\n] BEGIN DEAD;
<C>\n BEGIN INITIAL; {printf("Accepted\n");}

<D>b BEGIN G;
<D>a BEGIN E;
<D>[^ab\n] BEGIN DEAD;
<D>\n BEGIN INITIAL; {printf("Accepted\n");}

<E>b BEGIN F;
<E>a BEGIN B;
<E>[^ab\n] BEGIN DEAD;
<E>\n BEGIN INITIAL; {printf("Accepted\n");}

<F>b BEGIN G;
<F>a BEGIN E;
<F>[^ab\n] BEGIN DEAD;
<F>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<G>b BEGIN INITIAL;
<G>a BEGIN A;
<G>[^ab\n] BEGIN DEAD;
```

```
<G>\n BEGIN INITIAL; {printf("Accepted\n");}

<DEAD>[^\n] BEGIN DEAD;
<DEAD>\n BEGIN INITIAL; {printf("Invalid\n");}
%%
int yywrap()
{
 return 1;
}
int main()
{
 printf("Enter String\n");
 yylex();
 return 0;
}
```

Output:



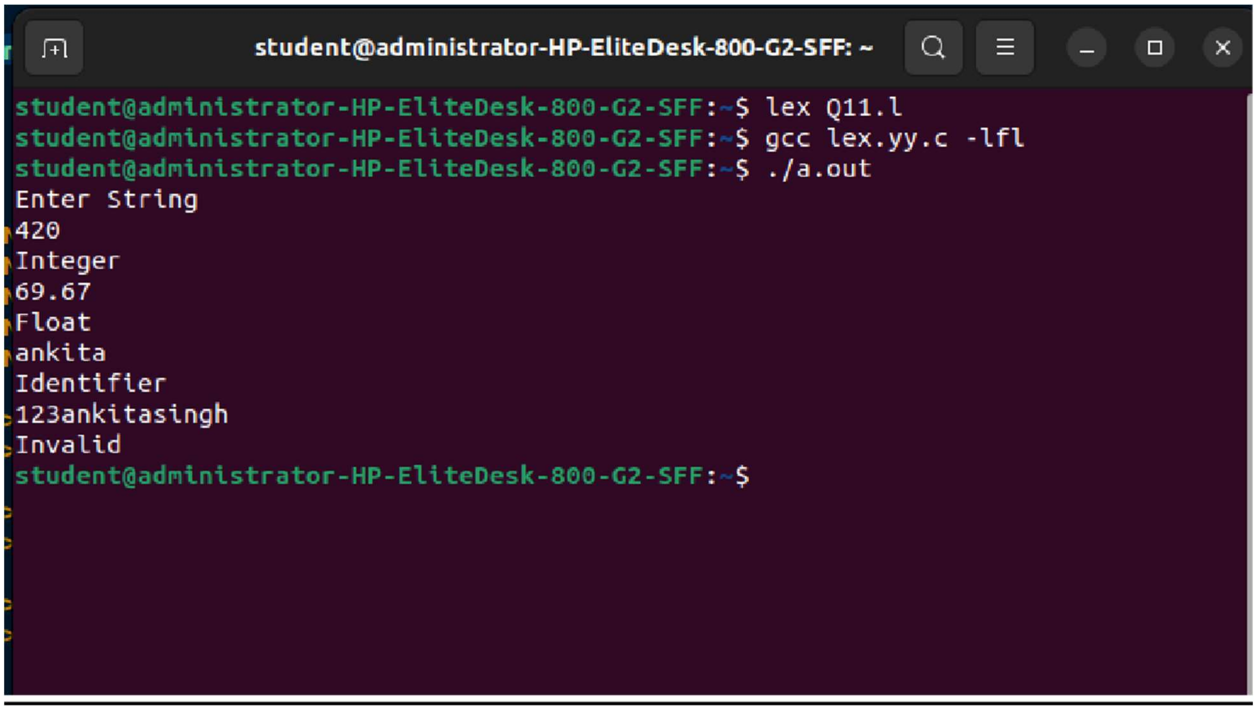
```
gehu@administrator-HP-EliteDesk-800-G2-SFF:~$ cd Desktop
gehu@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop$ flex p.l
gehu@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop$ gcc lex.yy.c
gehu@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop$ ./a.out
Enter String
aaba
Accepted
bba
Not Accepted
ca
Invalid
aaabbba
Not Accepted
```

Program 11: Design a DFA in LEX Code to identify and print integer and float constants and identifier.

Source Code

```
%{
#include<stdio.h>
%}
%s A B C DEAD
%%
<INITIAL>[0-9]+ BEGIN A;
<INITIAL>[0-9]+[.][0-9]+ BEGIN B;
<INITIAL>[A-Za-z_][A-Za-z0-9_]* BEGIN C;
<INITIAL>[^\n] BEGIN DEAD;
<INITIAL>\n BEGIN INITIAL; {printf("Not Accepted\n");}
<A>[^\n] BEGIN DEAD;
<A>\n BEGIN INITIAL; {printf("Integer\n");}
<B>[^\n] BEGIN DEAD;
<B>\n BEGIN INITIAL; {printf("Float\n");}
<C>[^\n] BEGIN DEAD;
<C>\n BEGIN INITIAL; {printf("Identifier\n");}
<DEAD>[^\n] BEGIN DEAD;
<DEAD>\n BEGIN INITIAL; {printf("Invalid\n");}
%%
int yywrap()
{return 1;}

int main()
{ printf("Enter
 String\n"); yylex();
 return 0 ; }
```

Output:



```
student@administrator-HP-EliteDesk-800-G2-SFF:~$ lex Q11.l
student@administrator-HP-EliteDesk-800-G2-SFF:~$ gcc lex.yy.c -lfl
student@administrator-HP-EliteDesk-800-G2-SFF:~$ ./a.out
Enter String
420
Integer
69.67
Float
ankita
Identifier
123ankitasingh
Invalid
student@administrator-HP-EliteDesk-800-G2-SFF:~$
```

Program 12 : Design a YAAC/ LEX Code to recognize valid arithmetic expression with operators +,-,*,/.

Source Code

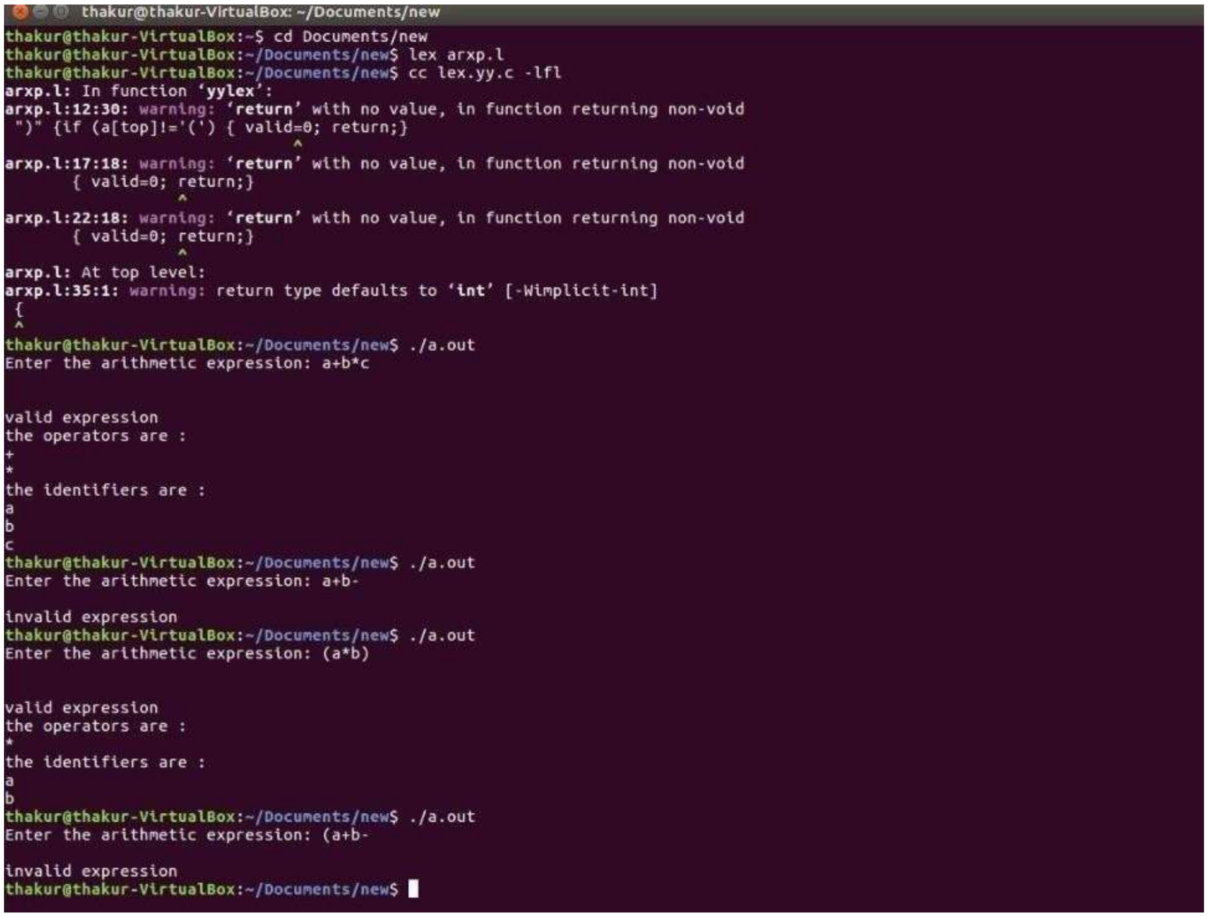
a.y file

```
 %{
   #include<stdio.h>
   int yylex(void);
   void yyerror(char *);
%}

%token digit
%%
S:S E '\n' {$$=$2;printf("output=%d\n",$$);}
   | ;
E:E '+' T {$$=$1+$3;}
 |E '-' T {$$=$1-$3;}
 |T {$$ = $1;}
 ;
T: T  '^' F {$$=$1*$3;}
  |T '/' F {$$=$1/$3;}
  |F {$$ = $1;}
  ;
F :digit {$$ = $1;}
%%
int main(){
   yyparse();
   return 0;
}
void yyerror(char *msg)
{
   printf("\n%s",msg);
   printf("\narithematic expression is invalid");}
}
```

b.l file

```
%{
   #include<stdio.h>
   int yylval;
   #include"y.tab.h"
%}
%%
[0-9]+ {yylval = atoi(yytext);return digit;}
[-+*/\n] return *yytext;
. ;
%%
int yywrap(void)
{
   return 1;}
```

Output:



```
thakur@thakur-VirtualBox: ~/Documents/new
thakur@thakur-VirtualBox:~$ cd Documents/new
thakur@thakur-VirtualBox:~/Documents/new$ lex arxp.l
thakur@thakur-VirtualBox:~/Documents/new$ cc lex.yy.c -lfl
arxp.l: In function ‘yylex’:
arxp.l:12:30: warning: ‘return’ with no value, in function returning non-void
 ")" {if (a[top]!='(') { valid=0; return;}
                              ^
arxp.l:17:18: warning: ‘return’ with no value, in function returning non-void
       { valid=0; return;}
                  ^
arxp.l:22:18: warning: ‘return’ with no value, in function returning non-void
       { valid=0; return;}
                  ^
arxp.l: At top level:
arxp.l:35:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 {
 ^
thakur@thakur-VirtualBox:~/Documents/new$ ./a.out
Enter the arithmetic expression: a+b*c


valid expression
the operators are :
+
*
the identifiers are :
a
b
c
thakur@thakur-VirtualBox:~/Documents/new$ ./a.out
Enter the arithmetic expression: a+b-

invalid expression
thakur@thakur-VirtualBox:~/Documents/new$ ./a.out
Enter the arithmetic expression: (a*b)


valid expression
the operators are :
*
the identifiers are :
a
b
thakur@thakur-VirtualBox:~/Documents/new$ ./a.out
Enter the arithmetic expression: (a+b-

invalid expression
thakur@thakur-VirtualBox:~/Documents/new$ 
```

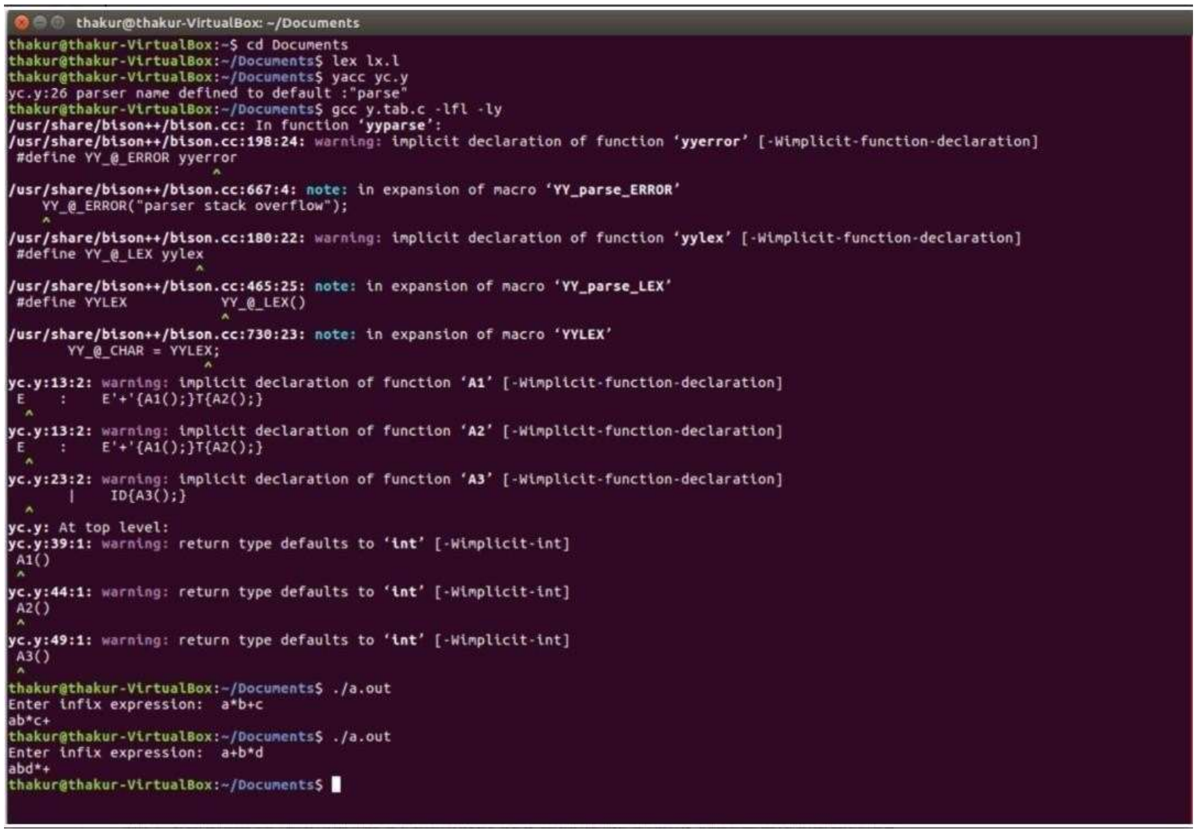
Program 13 : Design a YAAC/LEX code that translates infix expression to postfix expression.

Source Code

```
 ALPHA [A-Z a-z]
 DIGIT [0-9]
 %%
{ALPHA}({ALPHA}|{DIGIT})*      return ID;
{DIGIT}+                      {yylval=atoi(yytext); return ID;}
[\n \t]                       yyterminate();
.                             return yytext[0];
 %%

 %{
 #include <stdio.h>
 #include <stdlib.h>
 %}
%token      ID
 %left '+' '-'
 %left '*' '/'
 %left UMINUS

 %%

S      : E
E      : E'+'{A1();}T{A2();}
      | E'-'{A1();}T{A2();}
      | T
      ;
T      : T'*'{A1();}F{A2();}
      | T'/'{A1();}F{A2();}
      | F
      ;
 F : '('E{A2();}')'
      | '-'{A1();}F{A2();}
      | ID{A3();}
      ;

 %%

 #include "lex.yy.c"
 char st[100];
 int top=0;

 int main()
```

```
{ printf("Enter infix expression: ");
   yyparse(); printf("\n");
}

A1()
{ st[top++]=yytext[0];
}

A2()
{ printf("%c",st[--top]);
}

A3()
{ printf("%c",yytext[0]);
}
```

Output:



```
thakur@thakur-VirtualBox: ~/Documents
thakur@thakur-VirtualBox:~$ cd Documents
thakur@thakur-VirtualBox:~/Documents$ lex lx.l
thakur@thakur-VirtualBox:~/Documents$ yacc yc.y
yc.y:26 parser name defined to default :"parse"
thakur@thakur-VirtualBox:~/Documents$ gcc y.tab.c -lfl -ly
/usr/share/bison++/bison.cc: In function ‘yyparse’:
/usr/share/bison++/bison.cc:198:24: warning: implicit declaration of function ‘yyerror’ [-Wimplicit-function-declaration]
 #define YY_@_ERROR yyerror
                        ^
/usr/share/bison++/bison.cc:667:4: note: in expansion of macro ‘YY_parse_ERROR’
    YY_@_ERROR("parser stack overflow");
    ^
/usr/share/bison++/bison.cc:180:22: warning: implicit declaration of function ‘yylex’ [-Wimplicit-function-declaration]
 #define YY_@_LEX yylex
                      ^
/usr/share/bison++/bison.cc:465:25: note: in expansion of macro ‘YY_parse_LEX’
 #define YYLEX           YY_@_LEX()
                         ^
/usr/share/bison++/bison.cc:730:23: note: in expansion of macro ‘YYLEX’
       YY_@_CHAR = YYLEX;
                       ^
yc.y:13:2: warning: implicit declaration of function ‘A1’ [-Wimplicit-function-declaration]
 E    :    E'+'{A1();}T{A2();}
  ^
yc.y:13:2: warning: implicit declaration of function ‘A2’ [-Wimplicit-function-declaration]
 E    :    E'+'{A1();}T{A2();}
  ^
yc.y:23:2: warning: implicit declaration of function ‘A3’ [-Wimplicit-function-declaration]
       |    ID{A3();}
  ^
yc.y: At top level:
yc.y:39:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 A1()
 ^
yc.y:44:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 A2()
 ^
yc.y:49:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 A3()
 ^
thakur@thakur-VirtualBox:~/Documents$ ./a.out
Enter infix expression:  a*b+c
ab*c+
thakur@thakur-VirtualBox:~/Documents$ ./a.out
Enter infix expression:  a+b*d
abd*+
thakur@thakur-VirtualBox:~/Documents$ █
```

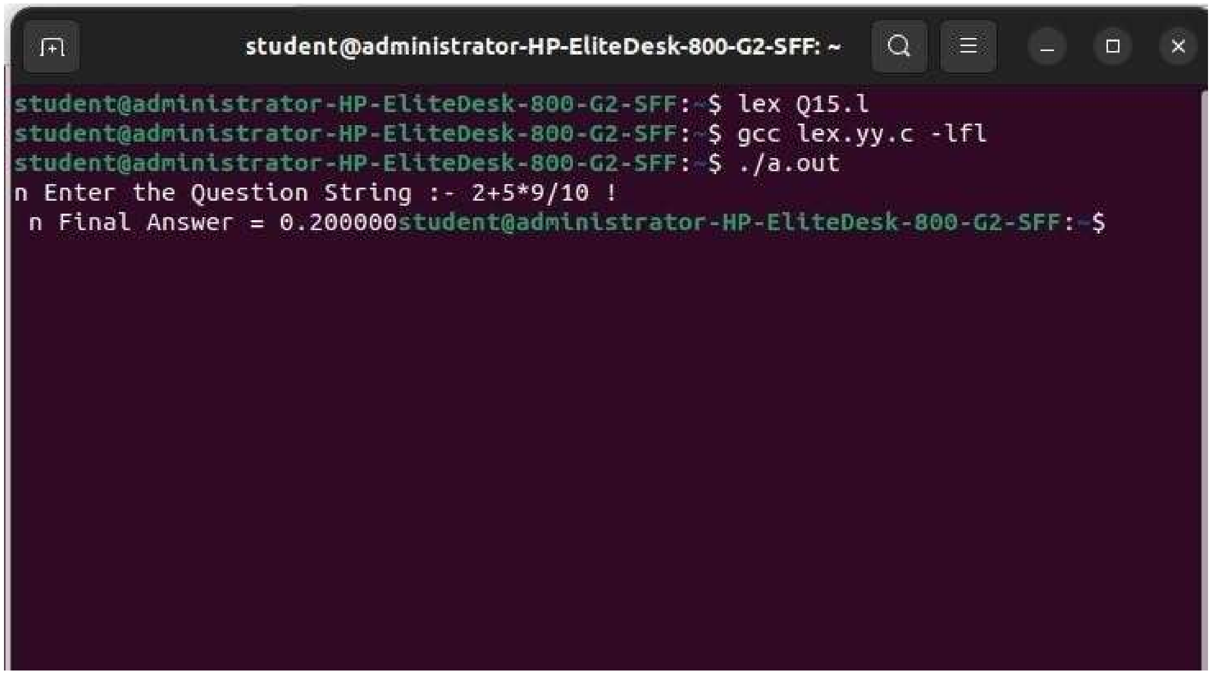
Program 14: Draw YAAC/LEX Code for Desk Calculator

Source Code

```
%{
#include<stdio.h>
float p,flag,answer;
char cc;
%}
digit [0-9]+ op
"+"|"-"|"*"|"/"
%%
{digit} {
p=atof(yytext)
; if(flag==0) {
answer=p;
flag=1; } else
{ switch(cc) {
case '+':answer=answer+p;
case '-':answer=answer-p; case
'*':answer=answer*p; case
'/':answer=answer/p;
}
}
}
{op} {
if(strcmp(yytext,"+")==0)
cc='+'; if(strcmp(yytext,"-
")==0) cc='-';
if(strcmp(yytext,"*")==0)
cc='*';
if(strcmp(yytext,"/")==0) cc='/';
}
```

```
! {printf("n Final Answer = %f",answer);exit(0);}
%%
int main() { flag=answer=0; printf("n
Enter the Question String :- "); yylex();
return(0); }
```

Output:



```
student@administrator-HP-EliteDesk-800-G2-SFF:~$ lex Q15.l
student@administrator-HP-EliteDesk-800-G2-SFF:~$ gcc lex.yy.c -lfl
student@administrator-HP-EliteDesk-800-G2-SFF:~$ ./a.out
n Enter the Question String :- 2+5*9/10 !
 n Final Answer = 0.200000student@administrator-HP-EliteDesk-800-G2-SFF:~$
```