Term Work

on

Compiler Design Lab

(**PCS-601**)

(2022-2023)

Submitted To:

Mr. Anirudha Prabhu

Assistant Professor

GEHU, D. Dun

Submitted By:

Pranjali Kothari

Student ID: 20012810

CSE-B- VI Sem

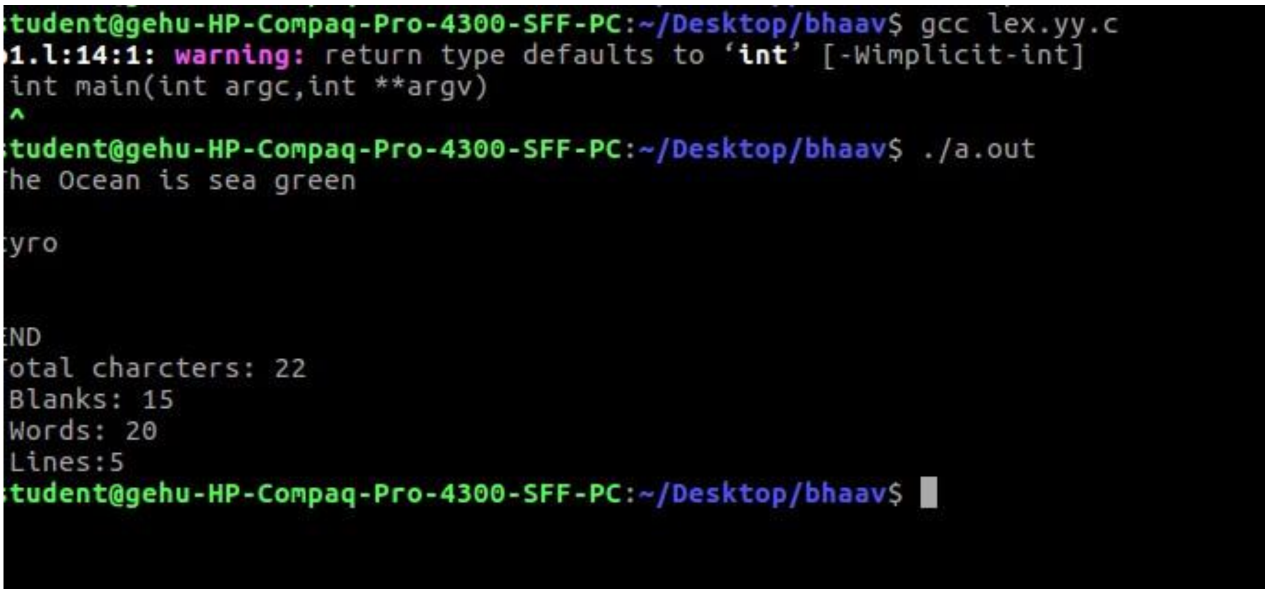
Session: 2022-2023

Program No. 01: Write a Lex Program that counts lines ,tabs ,spaces and characters.

SOURCE CODE:

```
%{
  int cc=0 , wc=0 , sc=0 , tc=0 , lc=0;
%}
%%
[a-zA-Z0-9][" "] {cc++;wc++;sc++;}
[a-zA-Z0-9][\t] {cc++;wc++;tc++;}
[a-zA-Z0-9][\n] {cc++;wc++;lc++;}
[a-zA-Z0-9] {cc++;}
[" "] {sc++;}
[\t] {tc++;}
[\n] {lc++;}
END return 0;
%%
yywrap(){ }
int main(int argc , int *argv)
{
yylex();
printf("Total characters: %d\n Blanks: %d\n words:%d\n Lines:%d\n tab space:%d\n" , cc,sc,wc,lc,tc);
  return 0;
}
```

OUTPUT



```
tudent@gehu-HP-Compaq-Pro-4300-SFF-PC:~/Desktop/bhaav$ gcc lex.yy.c
1.l:14:1: warning: return type defaults to 'int' [-Wimplicit-int]
 int main(int argc,int **argv)
 ^
tudent@gehu-HP-Compaq-Pro-4300-SFF-PC:~/Desktop/bhaav$ ./a.out
he Ocean is sea green

yro


ND
otal charcters: 22
 Blanks: 15
 Words: 20
 Lines:5
tudent@gehu-HP-Compaq-Pro-4300-SFF-PC:~/Desktop/bhaav$
```

Program No. 02: Write a Lex Program that identifies whether a Number is Integer or Float.

SOURCE CODE:

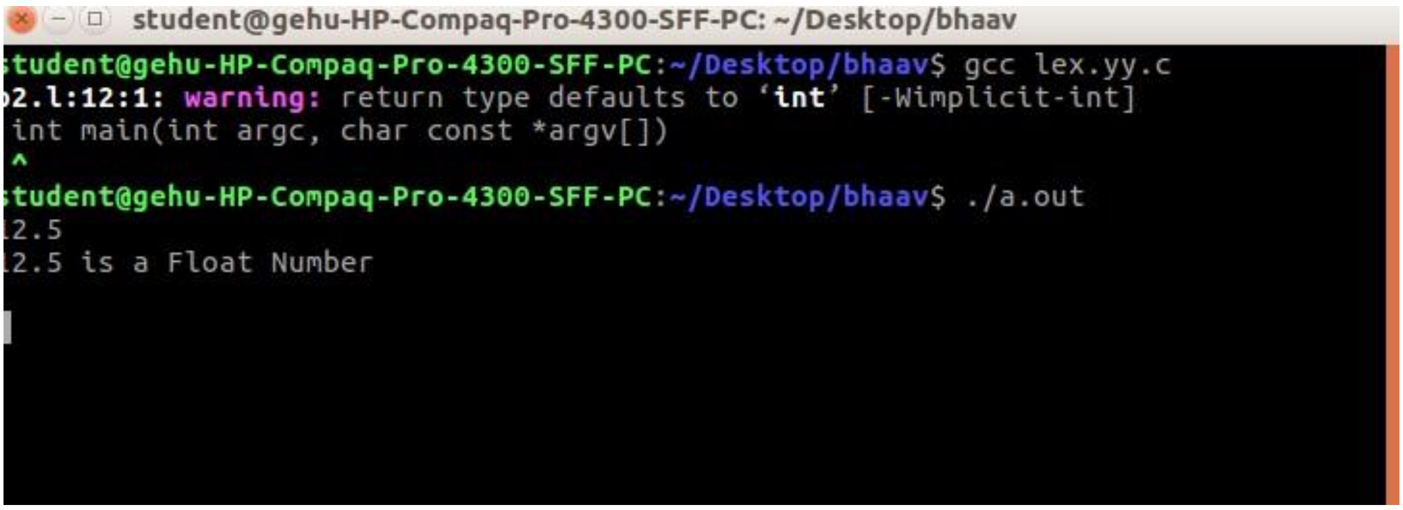
```
%{


%}


DIGIT [0-9]
%%
{DIGIT}* {ECHO;printf(" is a Integer Number\n");}
{DIGIT}*?\.{DIGIT}* {ECHO;printf(" is a Float Number\n");}
%%


yywrap(){}
int main(int argc, char const *argv[])
{
   yylex();
   return 0;
```

OUTPUT



```
student@gehu-HP-Compaq-Pro-4300-SFF-PC: ~/Desktop/bhaav
student@gehu-HP-Compaq-Pro-4300-SFF-PC:~/Desktop/bhaav$ gcc lex.yy.c
p2.l:12:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 int main(int argc, char const *argv[])
 ^
student@gehu-HP-Compaq-Pro-4300-SFF-PC:~/Desktop/bhaav$ ./a.out
12.5
12.5 is a Float Number
```

Program No. 03: Write a Lex Program that Identifies and Counts a valid identifier

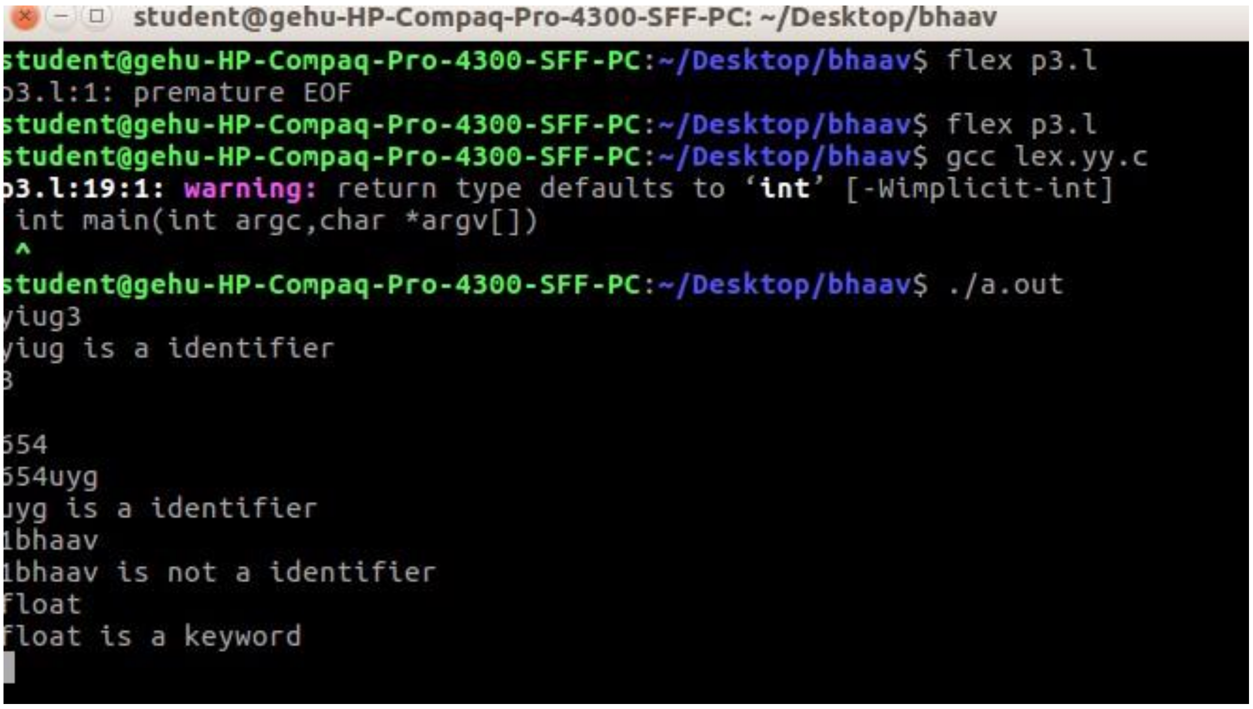
SOURCE CODE:

```
%{
int count=0;
%}

op [+-*/]
letter [a-zA-Z_]
digit [0-9]
id ({letter}+{digit}*)+
notid ({digit}+{letter}*)+

%%
[\t\n]+
("int")|("float")|("char")|("case")|("default")|("if")|("for")|("printf")|("scanf") {printf("%s is a
keyword\n",yytext);}
{id} {printf("%s is a identifier\n",yytext);count++;}
{notid} {printf("%s is not a identifier\n",yytext);}
{digit}* {printf("%s is not a identifier\n",yytext);}
[$] return 0;
%%
int yywrap(){return 1;}
int main()
{
yylex();
printf("Total Number of IDentifiers: %d\n",count);
return 0;
}
```

OUTPUT



student@gehu-HP-Compaq-Pro-4300-SFF-PC: ~/Desktop/bhaav

```
student@gehu-HP-Compaq-Pro-4300-SFF-PC:~/Desktop/bhaav$ flex p3.l
p3.l:1: premature EOF
student@gehu-HP-Compaq-Pro-4300-SFF-PC:~/Desktop/bhaav$ flex p3.l
student@gehu-HP-Compaq-Pro-4300-SFF-PC:~/Desktop/bhaav$ gcc lex.yy.c
p3.l:19:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 int main(int argc,char *argv[])
 ^
student@gehu-HP-Compaq-Pro-4300-SFF-PC:~/Desktop/bhaav$ ./a.out
yiug3
yiug is a identifier
3

654
654uyg
uyg is a identifier
1bhaav
1bhaav is not a identifier
float
float is a keyword
```

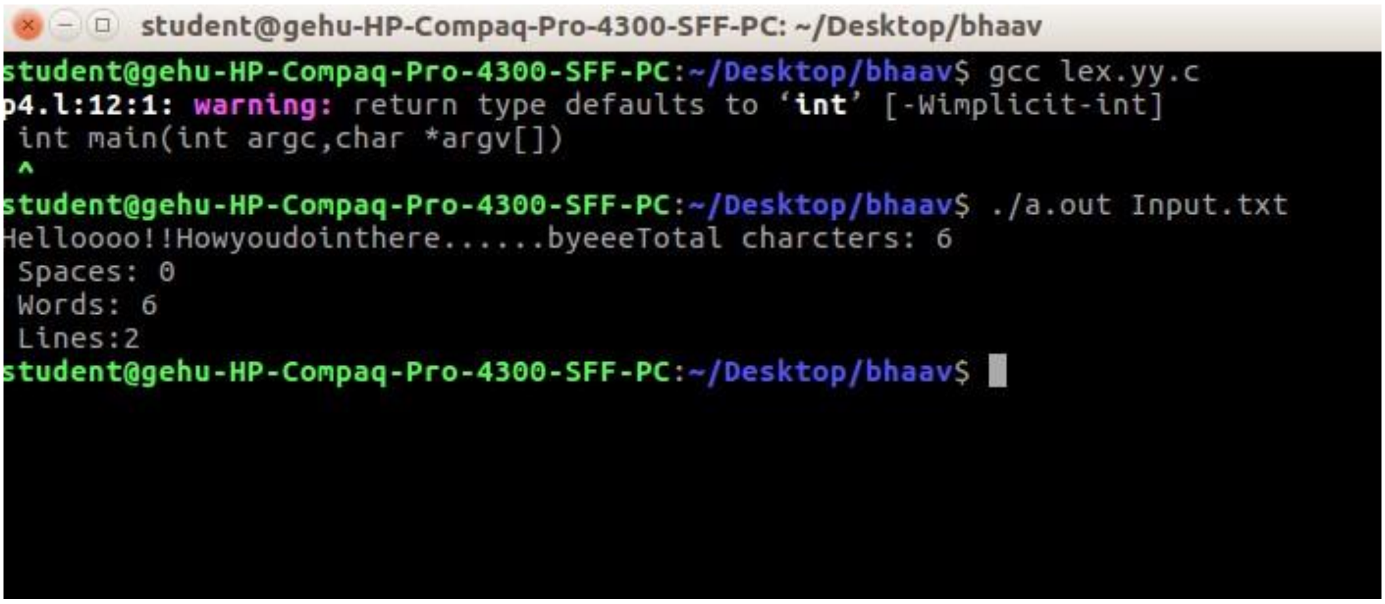
Program No. 04: Write a Lex Program that count totals characters ,white spaces, and words from a text file 'INPUT.txt'.

SOURCE CODE:

```
%{
   int cc=0 , wc=0 , sc=0 , tc=0 , lc=0;
%}


%%
[a-zA-Z0-9][" "] {cc++;wc++;sc++;}
[a-zA-Z0-9][\t] {cc++;wc++;tc++;}
[a-zA-Z0-9][\n] {cc++;wc++;lc++;}
[a-zA-Z0-9] {cc++;}
[" "] {sc++;}
[\t] {tc++;}
[\n] {lc++;}
EOF {return 0;}
%%
yywrap(){}
int main(int argc,char *argv[])
{
extern FILE *yyin;
yyin=fopen("Input.txt","r");
yylex();
printf("Total characters: %d\n Blanks: %d\n words:%d\n Lines:%d\n tab space:%d\n" , cc,sc,wc,lc,tc);
return 0;
}
```

OUTPUT



```
student@gehu-HP-Compaq-Pro-4300-SFF-PC: ~/Desktop/bhaav
student@gehu-HP-Compaq-Pro-4300-SFF-PC:~/Desktop/bhaav$ gcc lex.yy.c
p4.l:12:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 int main(int argc,char *argv[])
 ^
student@gehu-HP-Compaq-Pro-4300-SFF-PC:~/Desktop/bhaav$ ./a.out Input.txt
Helloooo!!Howyoudointhere......byeeeTotal charcters: 6
 Spaces: 0
 Words: 6
 Lines:2
student@gehu-HP-Compaq-Pro-4300-SFF-PC:~/Desktop/bhaav$ 
```

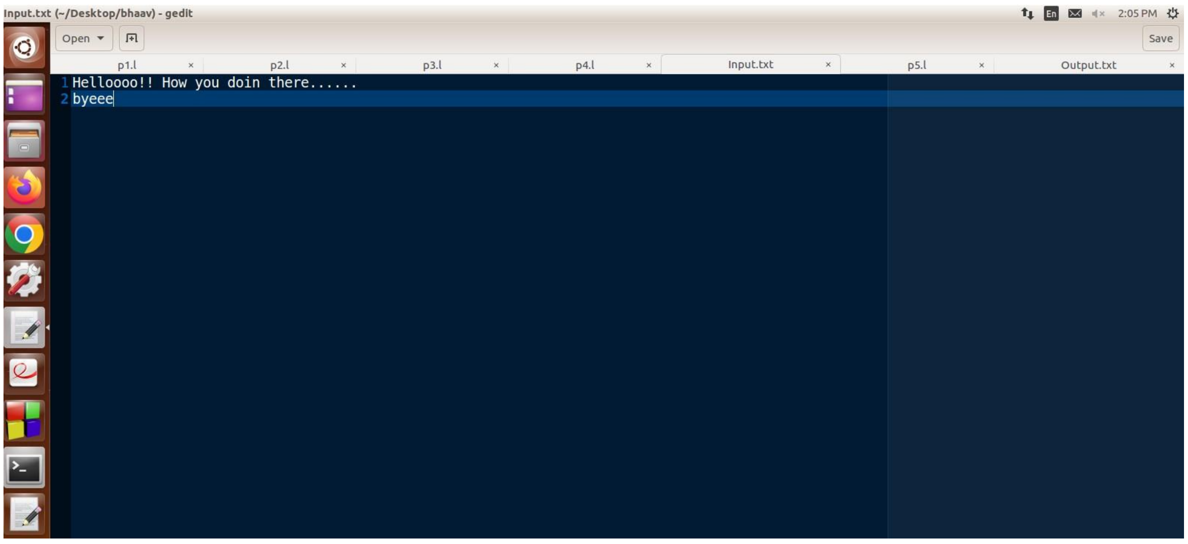
Program No. 05: Write a Lex Program that Replace tab space with a single space in a file Input.txt .

SOURCE CODE:

```
%{
%}
%%
[\t]+ fprintf(yyout," ");
. fprintf(yyout,"%s",yytext);
%%
yywrap(){}
main()
{
extern FILE *yyin,*yyout;
yyin=fopen("Input.txt","r");
yyout=fopen("Output.txt","w");
yylex();
return 0;
```

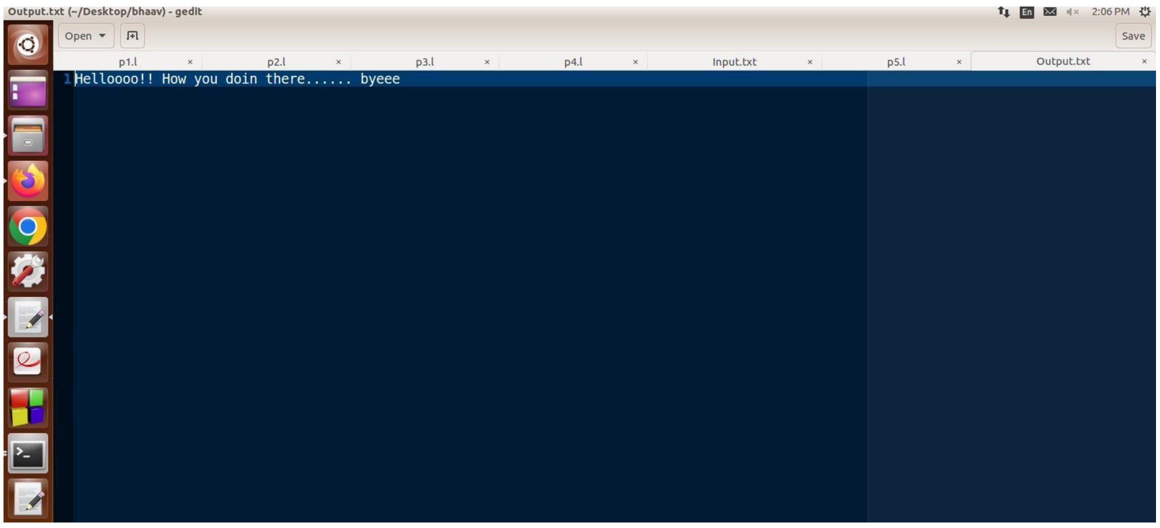
OUTPUT

INPUT.TXT



```
Helloooo!! How you doin there......
byeee
```

OUTPUT.TXT



```
Helloooo!! How you doin there...... byeee
```

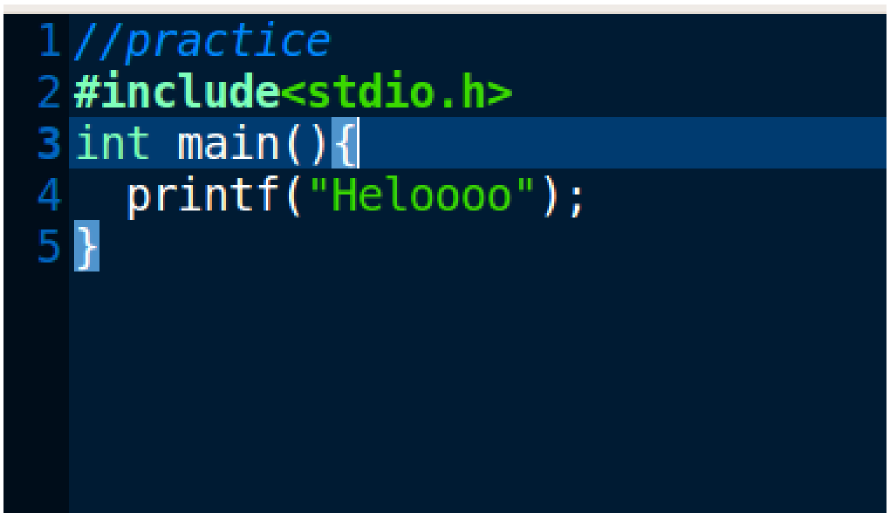
Program No. 06: Write a Lex Program that remove C comments (single and multi-line)

SOURCE CODE:

```
%{
%}
%%
"//"[^\n]* ;
"/*"([^*]|[*]+[^/])*[*]+"/" ;
. fprintf(yyout,"%s",yytext);
%%
yywrap(){}
main()
{
extern FILE *yyin,*yyout;
yyin=fopen("Input.c","r");
yyout=fopen("Output.c","w");
yylex();
return 0;
```

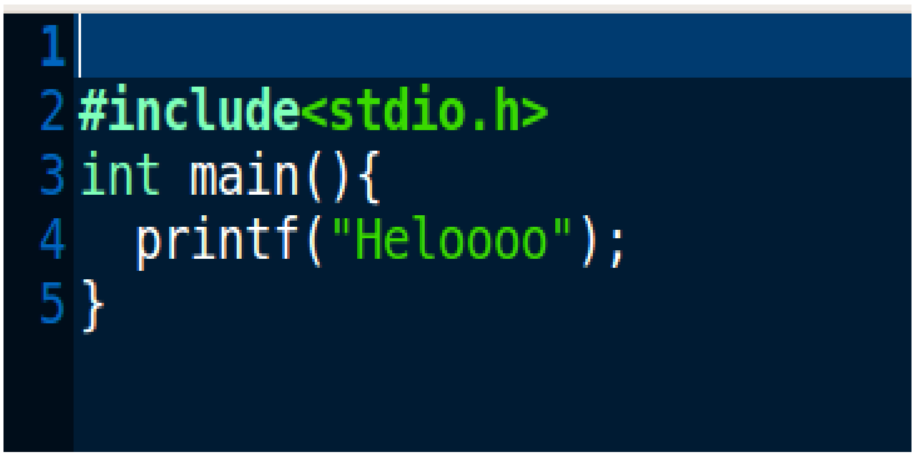
OUTPUT

INPUT.TXT



```c
//practice
#include<stdio.h>
int main(){
  printf("Heloooo");
}
```

OUTPUT.TXT



```c

#include<stdio.h>
int main(){
  printf("Heloooo");
}
```

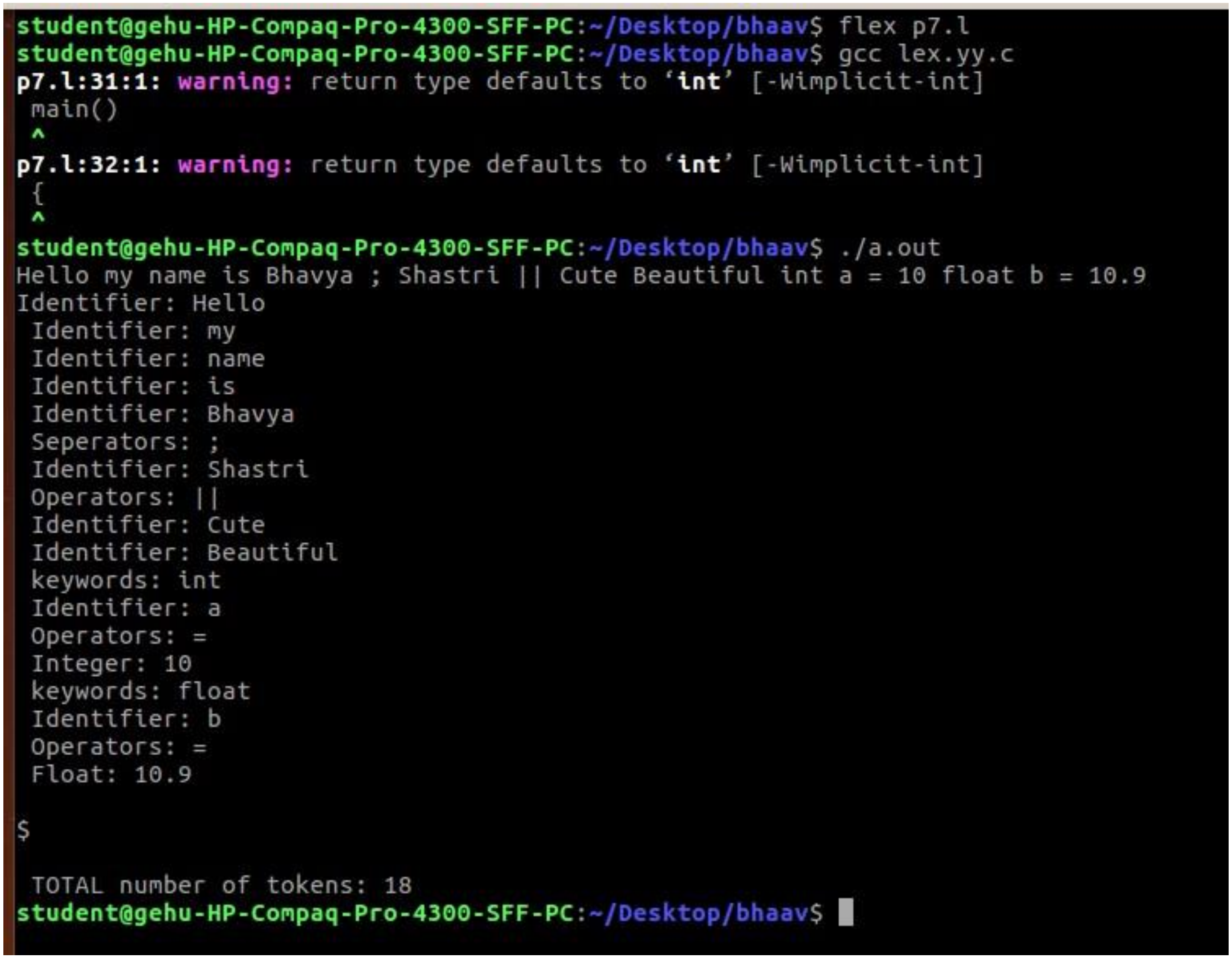
Program No. 07: Write a Lex Program for tokenizing(identify and print operators, separators, keywords, Identifiers) of following C-Program.

SOURCE CODE:

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

%%
"while"|"if"|"else" {n++;printf("keywords: %s\n",yytext);}
"int"|"float" {n++;printf("keywords: %s\n",yytext);}
[a-zA-Z_][a-zA-Z0-9_]* {n++;printf("Identifier: %s\n",yytext);}
"<="|"=="|"="|"++"|"-"|"*"|"+"|"("|")"|"||" {n++;printf("Operators: %s\n",yytext);}
"{"|"}"|";"|"," {n++;printf("Seperators: %s\n",yytext);}
-?[0-9]+"."[0-9]+ {n++;printf("Float: %s\n",yytext);}
-?[0-9]+ {n++;printf("Integer: %s\n",yytext);}
[$] return 0;
%%
yywrap(){}
main()
{
yylex();
printf("\n TOTAL number of tokens: %d\n",n);
}
```

OUTPUT



```
student@gehu-HP-Compaq-Pro-4300-SFF-PC:~/Desktop/bhaav$ flex p7.l
student@gehu-HP-Compaq-Pro-4300-SFF-PC:~/Desktop/bhaav$ gcc lex.yy.c
p7.l:31:1: warning: return type defaults to 'int' [-Wimplicit-int]
 main()
 ^
p7.l:32:1: warning: return type defaults to 'int' [-Wimplicit-int]
 {
 ^
student@gehu-HP-Compaq-Pro-4300-SFF-PC:~/Desktop/bhaav$ ./a.out
Hello my name is Bhavya ; Shastri || Cute Beautiful int a = 10 float b = 10.9
Identifier: Hello
 Identifier: my
 Identifier: name
 Identifier: is
 Identifier: Bhavya
 Seperators: ;
 Identifier: Shastri
 Operators: ||
 Identifier: Cute
 Identifier: Beautiful
 keywords: int
 Identifier: a
 Operators: =
 Integer: 10
 keywords: float
 Identifier: b
 Operators: =
 Float: 10.9

$

 TOTAL number of tokens: 18
student@gehu-HP-Compaq-Pro-4300-SFF-PC:~/Desktop/bhaav$ 
```

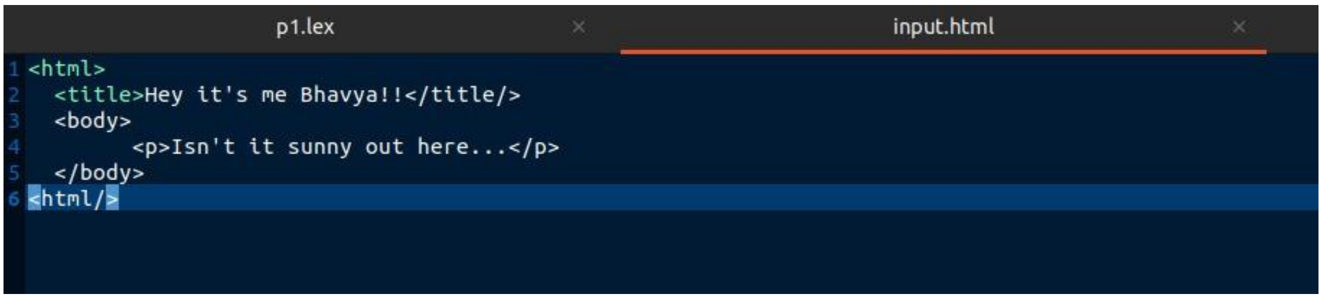
Program No. 08: Write a Lex Program that extract all html tags at runtime and store it in a file.

SOURCE CODE:

```
%{
%}
digit [0-9]
alpha [a-zA-Z0-9]
%%
"<"{alpha}*{digit}*">" fprintf(yyout,"%s",yytext);
"<"[/]{alpha}*{digit}*">" fprintf(yyout,"%s",yytext);
. ;
%%
int yywrap(){};
int main()
{
  extern FILE *yyin,*yyout;
  yyin=fopen("Input.html","r");
  yyout=fopen("Output.html","w");
  yylex();
  return 0;
}
```

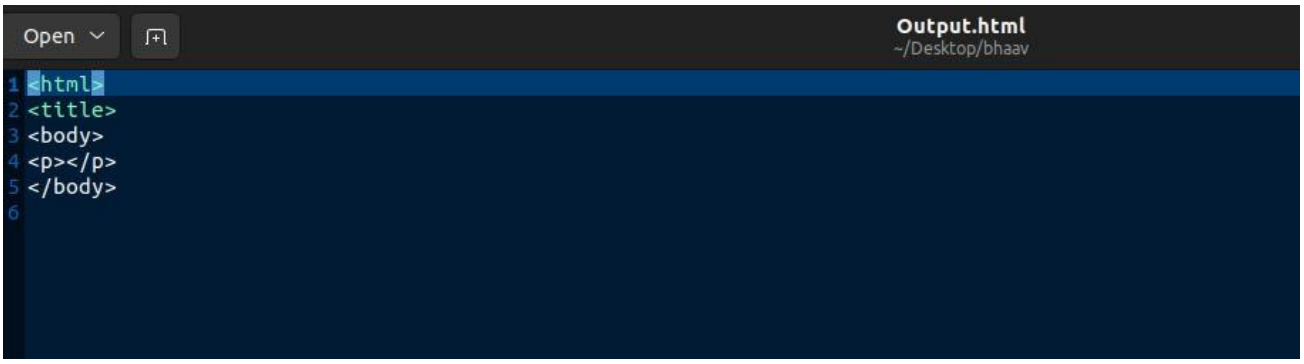
OUTPUT

INPUT.HTML



```
<html>
  <title>Hey it's me Bhavya!!</title/>
  <body>
        <p>Isn't it sunny out here...</p>
  </body>
<html/>
```

OUTPUT.HTML



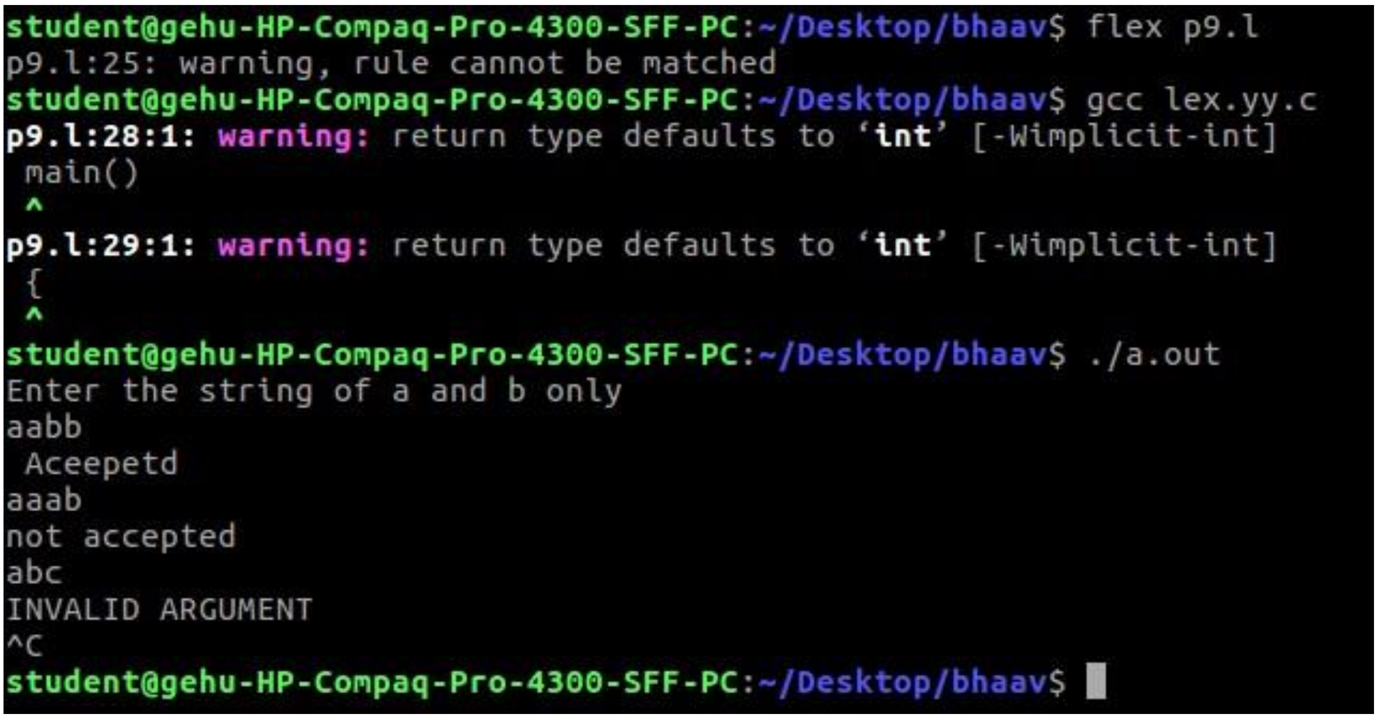
```
<html>
<title>
<body>
<p></p>
</body>
```

Program No. 09: Write a Lex Program using DFA to accept even number of input (Even number of a and even number of b)

SOURCE CODE:

```
{%
%}
%s A B C F
<INITIAL>\n {printf(" Aceepetd\n");} BEGIN INITIAL;
<INITIAL>a BEGIN A;
<INITIAL>b BEGIN B;
<A>a BEGIN INITIAL;
<A>b BEGIN C;
<A>\n BEGIN INITIAL; {printf("not accepted\n");}
<B>a BEGIN C;
<B>b BEGIN INITIAL;
<B>\n BEGIN INITIAL; {printf("not accepted\n");}
<C>a BEGIN B;
<C>b BEGIN A;
<C>\n BEGIN INITIAL; {printf("not accepted\n");}
<A>[^ab\n] BEGIN F;
<B>[^ab\n] BEGIN F;
<C>[^ab\n] BEGIN F;
<INITIAL>[^ab\n] BEGIN F;
<F>[^\n] BEGIN F;
<F>[\n] BEGIN INITIAL; {printf("INVALID ARGUMENT\n");}
. ;;
%%
yywrap(){}
main()
{
printf("Enter the string of a and b only\n");
yylex();
return 0;
}
```

OUTPUT



```
student@gehu-HP-Compaq-Pro-4300-SFF-PC:~/Desktop/bhaav$ flex p9.l
p9.l:25: warning, rule cannot be matched
student@gehu-HP-Compaq-Pro-4300-SFF-PC:~/Desktop/bhaav$ gcc lex.yy.c
p9.l:28:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 main()
 ^
p9.l:29:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 {
 ^
student@gehu-HP-Compaq-Pro-4300-SFF-PC:~/Desktop/bhaav$ ./a.out
Enter the string of a and b only
aabb
 Aceepetd
aaab
not accepted
abc
INVALID ARGUMENT
^C
student@gehu-HP-Compaq-Pro-4300-SFF-PC:~/Desktop/bhaav$ 
```

Program No. 10: Write a Lex Program using DFA which accepts string containing third last character ‘a’ over input alphabet {a, b}.

SOURCE CODE:

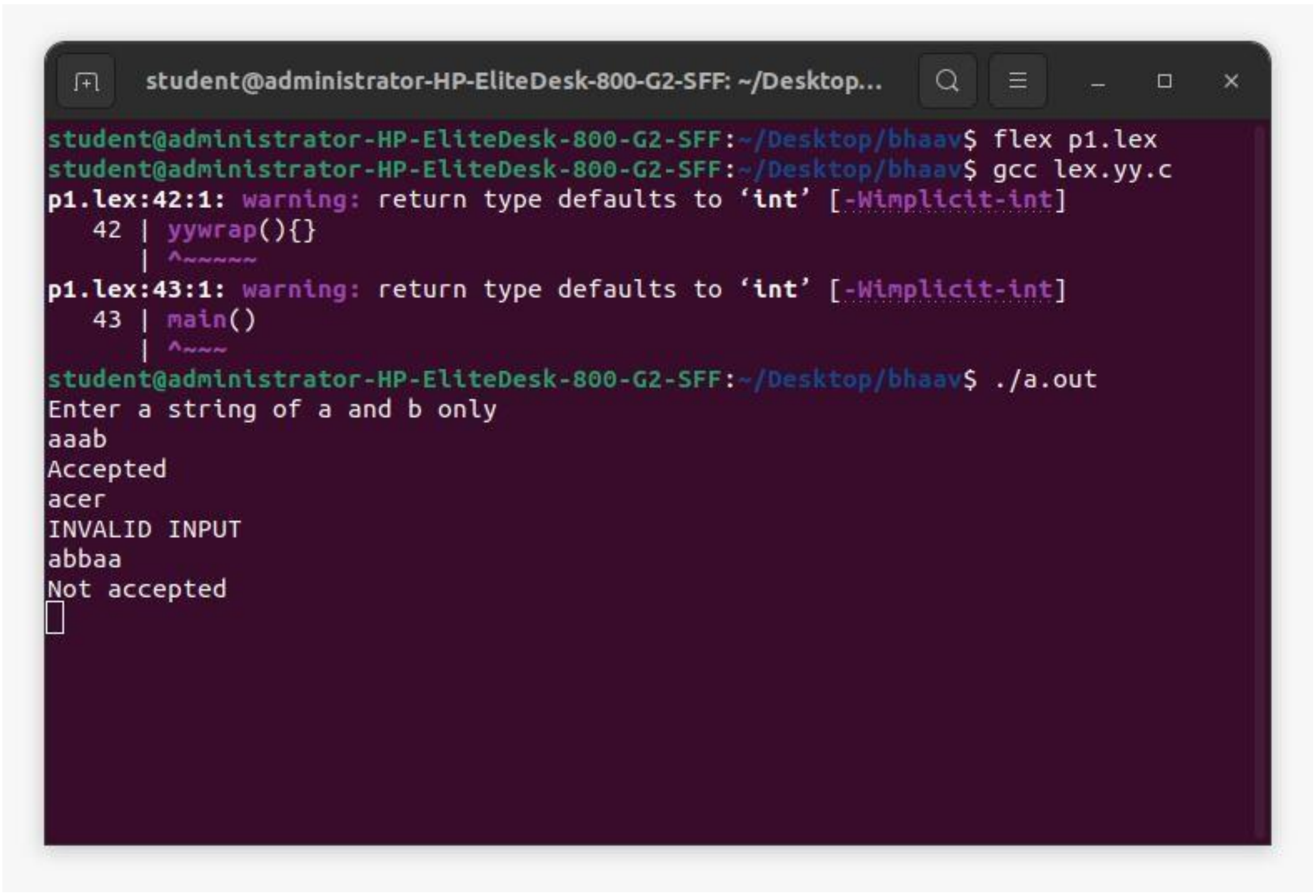
```
%{
%}

%s A B C D E F G H
%%
<INITIAL>a BEGIN A;
<INITIAL>b BEGIN INITIAL;
<A>a BEGIN D;
<A>b BEGIN B;
<B>a BEGIN E;
<B>b BEGIN C;
<C>a BEGIN A;
<C>b BEGIN INITIAL;
<D>a BEGIN G;
<D>b BEGIN F;
<E>a BEGIN A;
<E>b BEGIN B;
<F>a BEGIN E;
<F>b BEGIN C;
<G>a BEGIN G;
<G>b BEGIN F;
<INITIAL>\n BEGIN INITIAL; printf("Not accepted\n");
<A>\n BEGIN INITIAL;printf("Not accepted\n");
<B>\n BEGIN INITIAL;printf("Not accepted\n");
<C>\n BEGIN INITIAL;printf("Accepted\n");
<D>\n BEGIN INITIAL;printf("Not accepted\n");
<E>\n BEGIN INITIAL;printf("Accepted\n");
<F>\n BEGIN INITIAL;printf("Accepted\n");
<G>\n BEGIN INITIAL;printf("Accepted\n");
```

```
<INITIAL>[^ab\n] BEGIN H;
<A>[^ab\n] BEGIN H;
<B>[^ab\n] BEGIN H;
<C>[^ab\n] BEGIN H;
<D>[^ab\n] BEGIN H;
<E>[^ab\n] BEGIN H;
<F>[^ab\n] BEGIN H;
<G>[^ab\n] BEGIN H;
<H>[^\n] BEGIN H;
<H>[\n] BEGIN INITIAL;printf("INVALID INPUT\n");
%%

yywrap(){}
main()
{
printf("Enter a string of a and b only\n");
yylex();
return 0;
}
```

OUTPUT



student@administrator-HP-EliteDesk-800-G2-SFF: ~/Desktop...

```
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ flex p1.lex
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ gcc lex.yy.c
p1.lex:42:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   42 | yywrap(){}
      | ^~~~~~
p1.lex:43:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   43 | main()
      | ^~~~
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ ./a.out
Enter a string of a and b only
aaab
Accepted
acer
INVALID INPUT
abbaa
Not accepted
```

Program No. 11: Write a Lex Program using DFA to IDENTIFY and print Integer and float Constants and identifier.

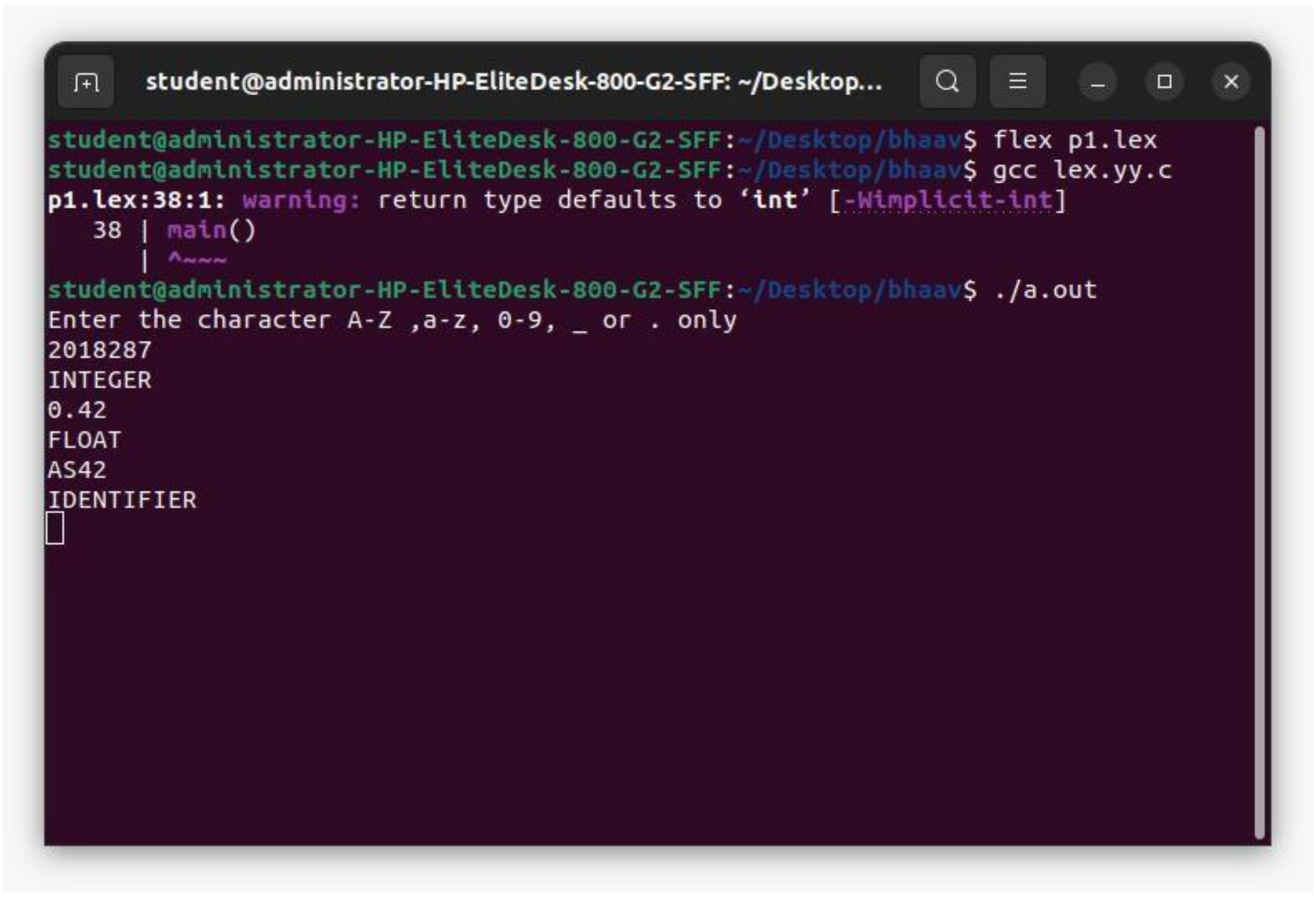
<u>SOURCE CODE:</u>

```
%{
%}
%s A B C D Y Z
%%
<INITIAL>[A-Za-z_] BEGIN B;
<INITIAL>[0-9] BEGIN A;
<INITIAL>[.] BEGIN Y;
<INITIAL>[^A_Za-z0-9_.\n] BEGIN Z;
<INITIAL>\n BEGIN INITIAL;printf("Not Accepted\n");
<A>[.] BEGIN C;
<A>[0-9] BEGIN A;
<A>[A-Za-z_] BEGIN Y;
<A>[^A_Za-z0-9_.\n] BEGIN Z;
<A>\n BEGIN INITIAL;printf("INTEGER\n");
<B>[A-Za-z_] BEGIN B;
<B>[0-9] BEGIN B;
<B>[.] BEGIN Y;
<B>[^A_Za-z0-9_.\n] BEGIN Z;
<B>\n BEGIN INITIAL; printf("IDENTIFIER\n");
<C>[0-9] BEGIN D;
<C>[.] BEGIN Y;
<C>[A-Za-z_] BEGIN Y;
<C>[^A_Za-z0-9_.\n] BEGIN Z;
<C>\n BEGIN INITIAL;printf("Not Accepted\n");
<D>[0-9] BEGIN D;
<D>[.] BEGIN Y;
<D>[A-Za-z_] BEGIN Y;
```

```
<D>[^A_Za-z0-9_.\n] BEGIN Z;
<D>\n BEGIN INITIAL; printf("FLOAT\n");
<Y>[A_Za-z0-9_.] BEGIN Y;
<Y>[^A_Za-z0-9_.\n] BEGIN Z;
<Y>[\n] BEGIN INITIAL;printf("Not Accpeted\n");
<Z>[^\n] BEGIN Z;
<Z>[\n] BEGIN INITIAL;printf("Invalid Input\n");
%%

int yywrap(){}
main()
{
printf("Enter the character A-Z ,a-z, 0-9, _ or . only\n");
yylex();
return 0;
}
```

OUTPUT



```
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ flex p1.lex
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ gcc lex.yy.c
p1.lex:38:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   38 | main()
      | ^~~~
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ ./a.out
Enter the character A-Z ,a-z, 0-9, _ or . only
2018287
INTEGER
0.42
FLOAT
AS42
IDENTIFIER
```

YACC

Program No. 12: Write a Lex-Yacc Program to recognize valid arithmetic expression with operators (+ , -, *, /).

SOURCE CODE:

Lex File

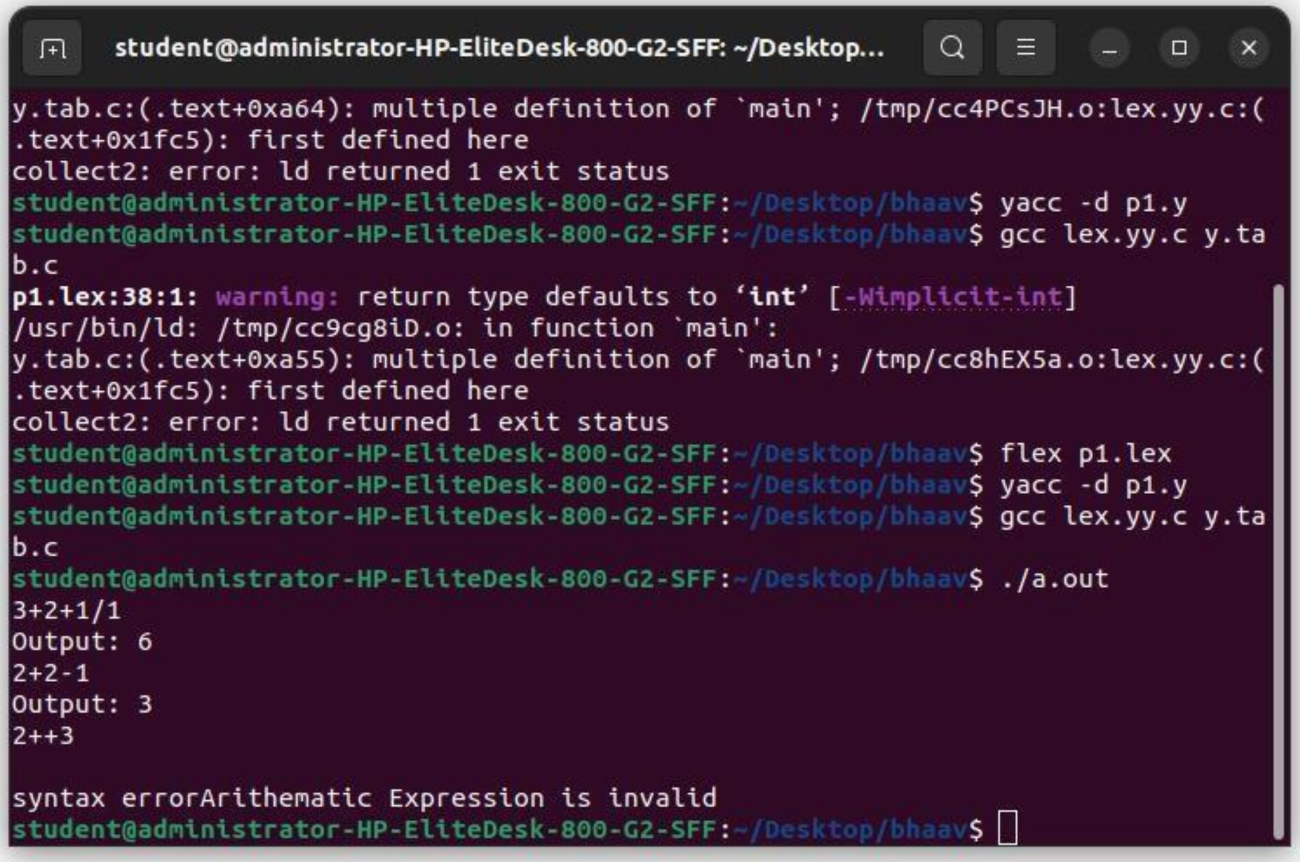
```
%{
#include<stdio.h>
#include "y.tab.h"
%}
%%
[0-9]+ {yylval=atoi(yytext);return NUMBER;}
[-+*/\n] return *yytext;
. ;
%%
int yywrap(){
return 1;
}
```

Yacc File

```
%{
  #include<stdio.h>
  #include<stdlib.h>
  void yyerror(char *);
  int yylex();
  int yywrap();
%}
%token NUMBER
%left '+' '-'
%left '*' '/'
%%
```

```
S: S E '\n' {$$=$2; printf("Output: %d\n",$$);}
 | ;
E: E'+'E {$$=$1+$3;}
  | E'-'E {$$=$1-$3;}
  | E'*'E {$$=$1*$3;}
  | E'/'E {if($3==0) {yyerror("Error...Division by Zero\n");}
            else {$$=$1/$3;}}
  |NUMBER {$$=$1;}
  ;
%%
void yyerror(char *msg)
{
printf("\n%s",msg);
printf("Arithematic Expression is invalid\n");
exit(0);
}
int yywrap(){ return 1;}
int main(){
yyparse();
return 0;
}
```

OUTPUT



```
y.tab.c:(.text+0xa64): multiple definition of `main'; /tmp/cc4PCsJH.o:lex.yy.c:(
.text+0x1fc5): first defined here
collect2: error: ld returned 1 exit status
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ yacc -d p1.y
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ gcc lex.yy.c y.ta
b.c
p1.lex:38:1: warning: return type defaults to 'int' [-Wimplicit-int]
/usr/bin/ld: /tmp/cc9cg8iD.o: in function `main':
y.tab.c:(.text+0xa55): multiple definition of `main'; /tmp/cc8hEX5a.o:lex.yy.c:(
.text+0x1fc5): first defined here
collect2: error: ld returned 1 exit status
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ flex p1.lex
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ yacc -d p1.y
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ gcc lex.yy.c y.ta
b.c
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ ./a.out
3+2+1/1
Output: 6
2+2-1
Output: 3
2++3

syntax errorArithematic Expression is invalid
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ 
```

Program No. 13: Write a Lex-Yacc Program to recognize valid arithmetic expression involving operators (+ , -, *, /) within CFG.

SOURCE CODE:

Lex File

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

Yacc File

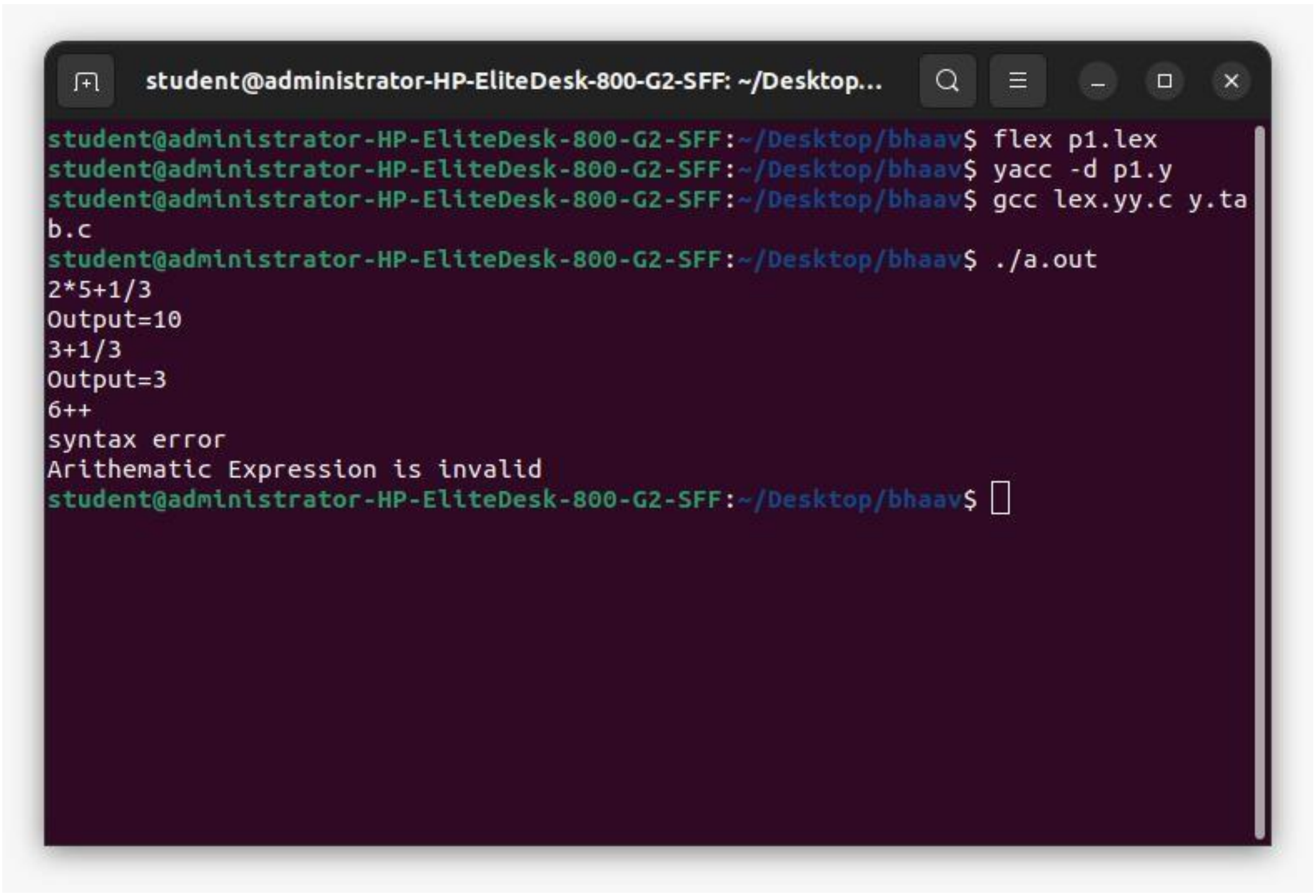
```
%{
#include<stdio.h>
#include<stdlib.h>
int yylex();
void yyerror(char *);
%}
%token digit



%%
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 {if($3==0) yyerror("Division by Zero!!") ;
        else $$=$1/$3;}
|F {$$=$1;}
 ;
F:digit {$$=$1;}
%%
int main(){
yyparse();
return 0;
}
void yyerror(char *msg)
{
printf("%s\n",msg);
printf("Arithematic Expression is invalid\n");
exit(1);
}
```

OUTPUT



```
student@administrator-HP-EliteDesk-800-G2-SFF: ~/Desktop...
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ flex p1.lex
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ yacc -d p1.y
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ gcc lex.yy.c y.ta
b.c
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ ./a.out
2*5+1/3
Output=10
3+1/3
Output=3
6++
syntax error
Arithematic Expression is invalid
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$
```

Program No. 14: Write a Lex-Yacc Program to convert infix expression to postfix expression.

SOURCE CODE:

Lex File

```
%{
#include "y.tab.h"
extern int yyval;
%}

%%
[0-9]+ {yylval=atoi(yytext); return NUM;}
\n return 0;
. return *yytext;
%%
int yywrap(){
return 1;
}
```

Yacc File

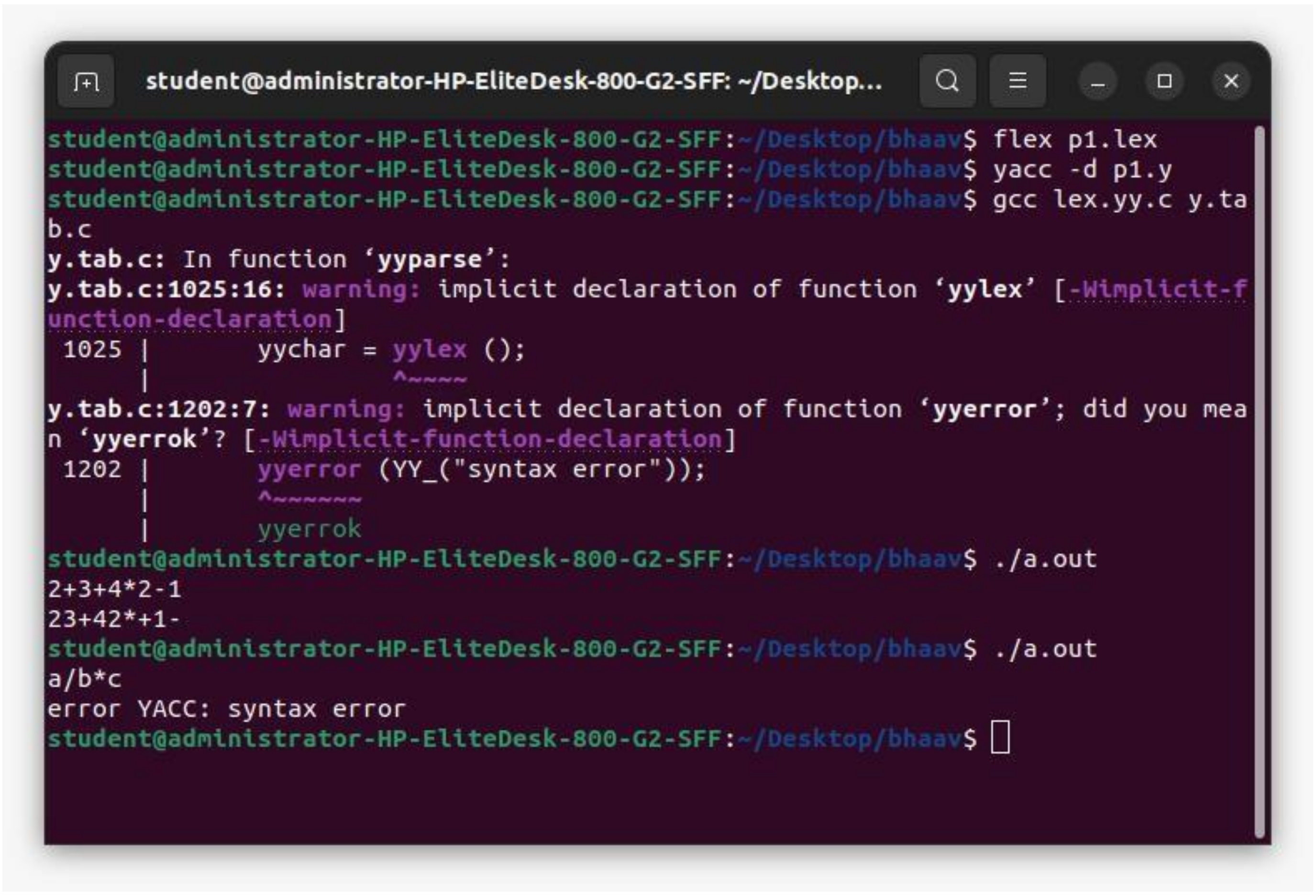
```
%{
#include<stdio.h>
%}

%token NUM
%left '+' '-'
%left '*' '/'
%right NEGATIVE
%%
S: E {printf("\n");}
;
E: E '+' E {printf("+");}
   | E '*' E {printf("*");}
```

```
    | E '-' E {printf("-");}
    |E '/' E {printf("/");}
    | '(' E ')'
    | '-' E %prec NEGATIVE {printf("-");}
    | NUM {printf("%d",yylval);}
    ;
%%

int main()
{
  yyparse();
}
int yyerror(char *msg)
{
   return printf("error YACC: %s\n", msg);
}
```

OUTPUT



```
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ flex p1.lex
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ yacc -d p1.y
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ gcc lex.yy.c y.ta
b.c
y.tab.c: In function ‘yyparse’:
y.tab.c:1025:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-f
unction-declaration]
 1025 |       yychar = yylex ();
      |                ^~~~~
y.tab.c:1202:7: warning: implicit declaration of function ‘yyerror’; did you mea
n ‘yyerrok’? [-Wimplicit-function-declaration]
 1202 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
      |       yyerrok
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ ./a.out
2+3+4*2-1
23+42*+1-
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ ./a.out
a/b*c
error YACC: syntax error
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$
```

Program No. 15: Write a Lex-Yacc Program to design a Desk Calculator.

SOURCE CODE:

Lex File

```
%{
#include  "y.tab.h"
#include<stdio.h>
#include<stdlib.h>
%}

%%
[a-z] {yylval=*yytext-'a'; return id;}
[0-9]+ {yylval=atoi(yytext); return digit;}
[-+()=/*\n] {return *yytext;}
[ \t] ;
. {printf("Invalid character\n"); exit(0);}
%%
int yywrap(){
return 1;
}
```

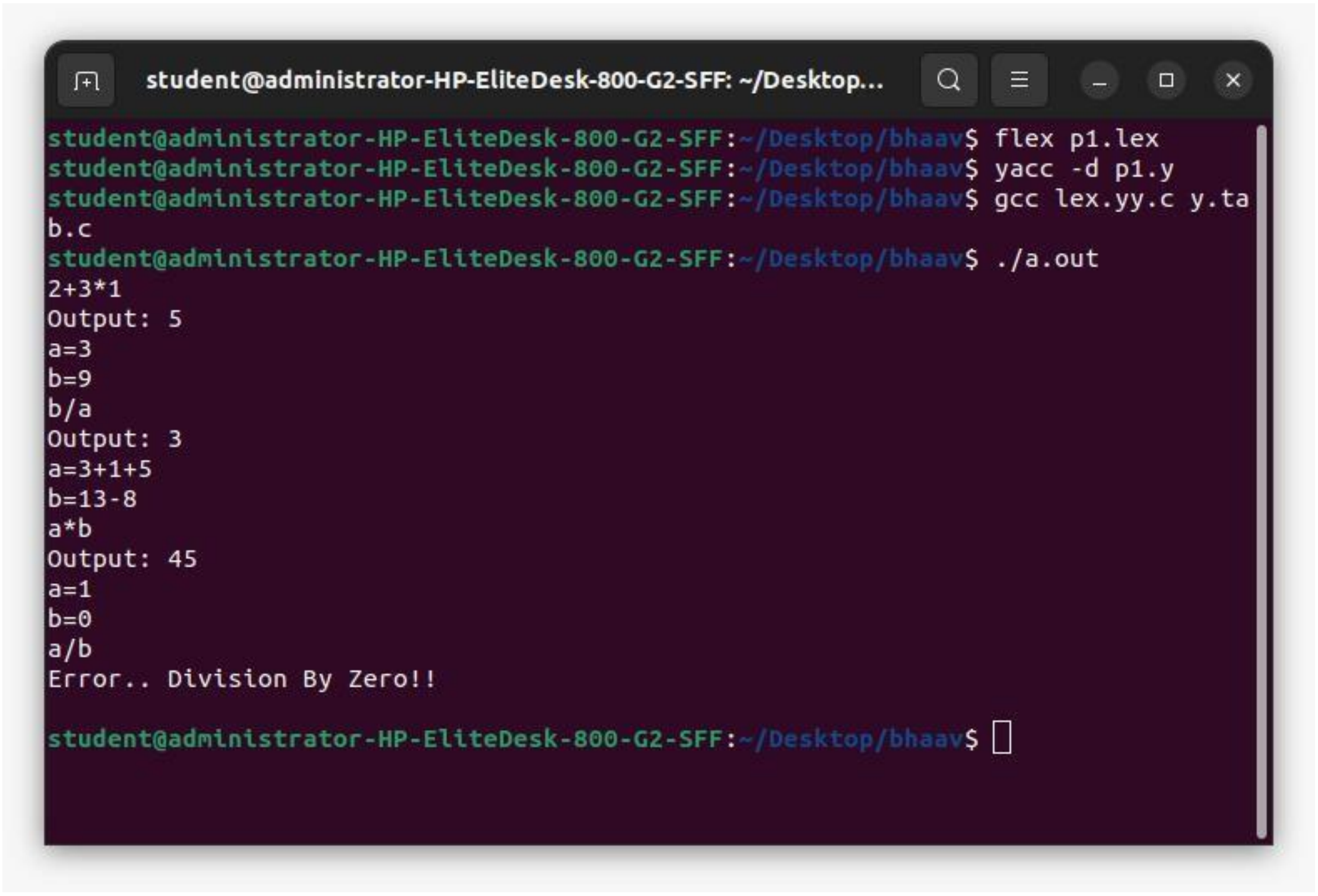
Yacc File

```
%{
#include<stdio.h>
#include<stdlib.h>
void yyerror(char *);
int yylex();
int sym[26]={0};
%}
%token id digit
%left '+' '-'
%left '*' '/'
%%
```

```
P: P S '\n'
| ;
S: E  {printf("Output: %d\n",$1);}
  | id '=' E {sym[$1]=$3;}
E: digit {$$=$1;}
    |id {$$=sym[$1];}
    |E '+' E {$$=$1+$3;}
    |E '-' E {$$=$1-$3;}
    |E '*' E {$$=$1*$3;}
    |E '/' E {if($3) $$=$1/$3;
              else{yyerror("Error.. Division By Zero!!\n");}}
    | '(' E ')'  {$$=$2;}
    ;
%%

int main()
{
  yyparse();
  return 0;
}
void yyerror(char *msg)
{
   fprintf(stderr,"%s\n", msg);
   exit(0);
}
```

OUTPUT



```
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ flex p1.lex
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ yacc -d p1.y
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ gcc lex.yy.c y.ta
b.c
student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ ./a.out
2+3*1
Output: 5
a=3
b=9
b/a
Output: 3
a=3+1+5
b=13-8
a*b
Output: 45
a=1
b=0
a/b
Error.. Division By Zero!!

student@administrator-HP-EliteDesk-800-G2-SFF:~/Desktop/bhaav$ 
```