Ramanujan College

University of Delhi

System Programming

Practical File

College Roll: 20201424

Shivank Chaudhary
BSc(H.) Comp Sci
20020570031

Practicals

1. Write a Lex program to count the number of lines and characters in the input file.

2. Write a Lex program that implements the Caesar cipher: it replaces every letter with the one three letters after in in alphabetical order, wrapping around at Z. e.g., a is replaced by d, b bye, and so on z by c.

3. Write a Lex program that finds the longest word (defined as a contiguous string of upper- and lower-case letters) in the input.

4. Write a Lex program that distinguishes keywords, integers, floats, identifiers, operators, and comments in any simple programming language.

5. Write a Lex program to count the number of identifiers in a C file.

6. Write a Lex program to count the number of words, characters, blank spaces and lines in a C file.

7. Write a Lex specification program that generates a C program which takes a string “abcd” and prints the following output. abcd abc ab a

8. A program in Lex to recognize a valid arithmetic expression.

9.Write a YACC program to find the validity of a given expression (for operators + - * and /)

10. A Program in YACC which recognizes a valid variable which starts with letter followed by a digit. The letter should be in lowercase only.

11. A Program in YACC to evaluate an expression (simple calculator program for addition and subtraction, multiplication, division).

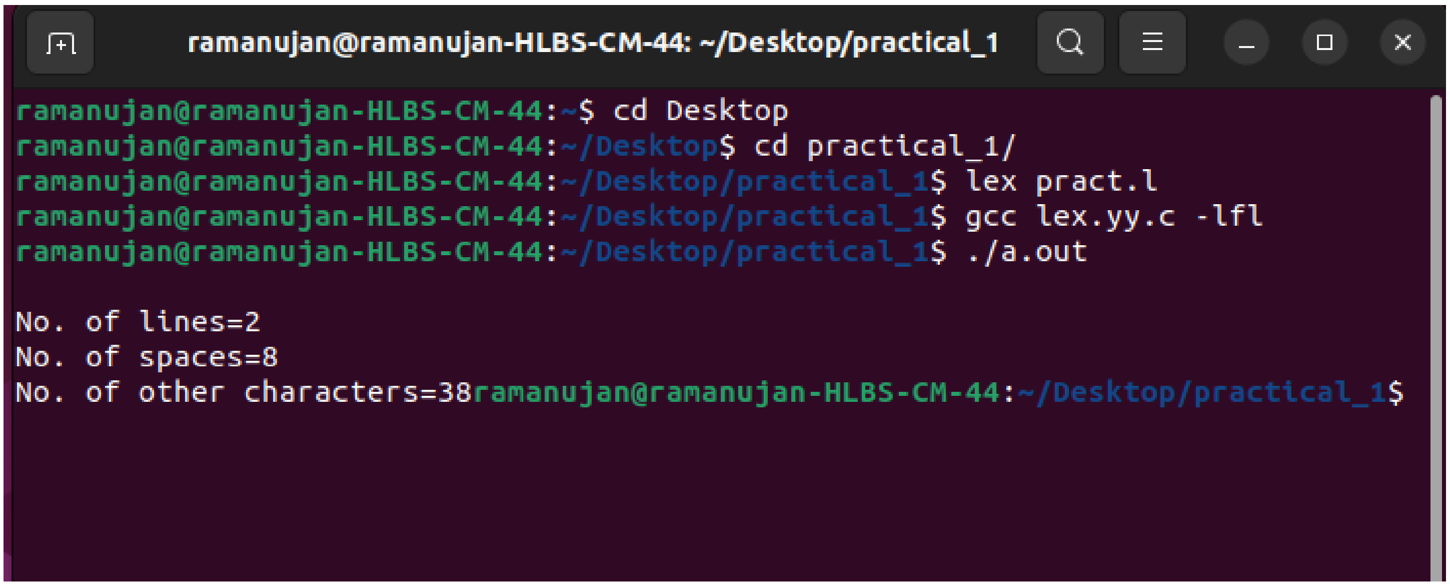
12. Program in YACC to recognize the strings “ab”, “aabb”,” aaabbb” ... of the language ($a\ n\ b\ n$, n>=1).

13. Program in YACC to recognize the language ($a\ nb$, n>=10). (Output to say input is valid or not

Practical 1

1. Write a Lex program to count the number of lines and characters in the input file.

```
%{
#include<stdio.h>
int lc=0, sc=0, tc=0, ch=0; /*Global variables*/
%}
/*Rule Section*/
%%
\n lc++; //line counter
([ ])+ sc++; //space counter
\t tc++; //tab counter
. ch++; //characters counter
%%
int main()
{
// The function that starts the analysis
yyin=fopen("abc.txt","r");
yylex();
printf("\nNo. of lines=%d", lc);
printf("\nNo. of spaces=%d", sc);
printf("\nNo. of other characters=%d", ch);
}
```



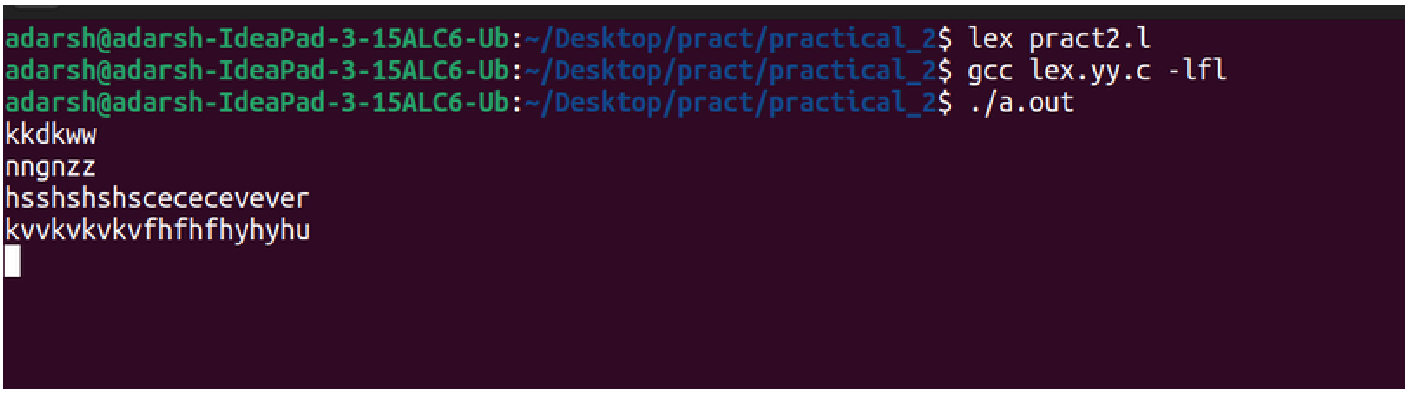
```
ramanujan@ramanujan-HLBS-CM-44:~$ cd Desktop
ramanujan@ramanujan-HLBS-CM-44:~/Desktop$ cd practical_1/
ramanujan@ramanujan-HLBS-CM-44:~/Desktop/practical_1$ lex pract.l
ramanujan@ramanujan-HLBS-CM-44:~/Desktop/practical_1$ gcc lex.yy.c -lfl
ramanujan@ramanujan-HLBS-CM-44:~/Desktop/practical_1$ ./a.out

No. of lines=2
No. of spaces=8
No. of other characters=38ramanujan@ramanujan-HLBS-CM-44:~/Desktop/practical_1$
```

Practical 2

2. Write a Lex program that implements the Caesar cipher: it replaces every letter with the one three letters after in in alphabetical order, wrapping around at Z. e.g., a is replaced by d, b bye, and so on z by c.

```
%{
%}
%%
[A-Wa-w] {printf("%c",yytext[0]+3);}
[X-Zx-z] {printf("%c",yytext[0]-23);}
%%
int main()
{
//yyin=fopen("bbc.txt","r");
//yyout=fopen("kbc.txt","w");
yylex();
}
```

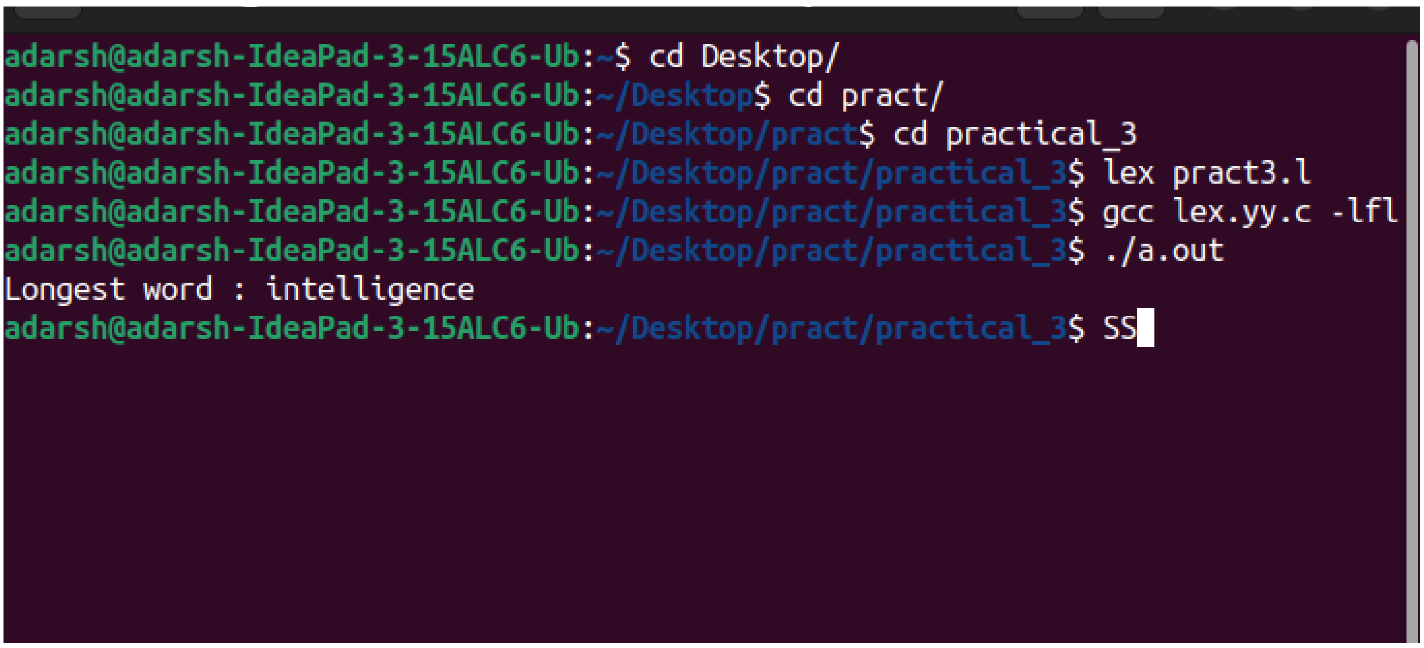


```
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_2$ lex pract2.l
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_2$ gcc lex.yy.c -lfl
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_2$ ./a.out
kkdkww
nngnzz
hsshshshscececevever
kvvkvkvkvfhfhfhyhyhu
```

Practical 3

3. Write a Lex program that finds the longest word (defined as a contiguous string of upper- and lower-case letters) in the input.

```
%{
#include<stdio.h>
#include<strings.h>
// initialising length
int length=0;
// char array for storing longest word
char longestword[50];
%}
%%
[A-Za-z0-9]+ { if (yyleng > length) {
length=yyleng;
 // strcpy function to copy current word in yytxt in longest
strcpy(longestword,yytext);
}
}
"." return 1;
%%
int main()
{
yyin=fopen("tbc.txt","r");
yylex();
printf("Longest word : %s\n",longestword);
//printf("Length of Longest word : %s\n",length);
return 0;
}
int yywrap(){
 return 1;
}
```

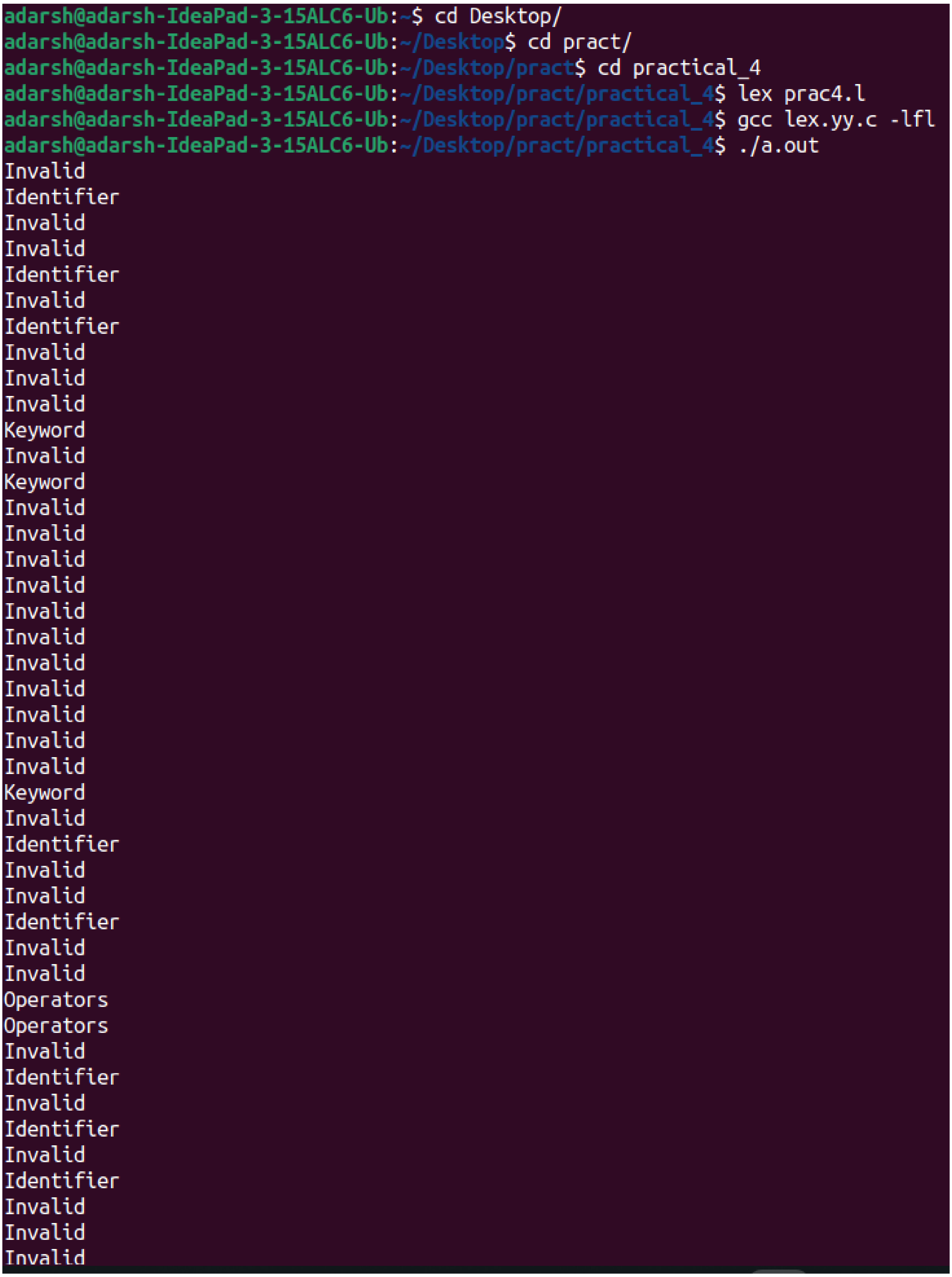


```
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~$ cd Desktop/
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop$ cd pract/
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract$ cd practical_3
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_3$ lex pract3.l
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_3$ gcc lex.yy.c -lfl
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_3$ ./a.out
Longest word : intelligence
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_3$ SS
```

Practical 4

4. Write a Lex program that distinguishes keywords, integers, floats, identifiers, operators, and comments in any simple programming language.

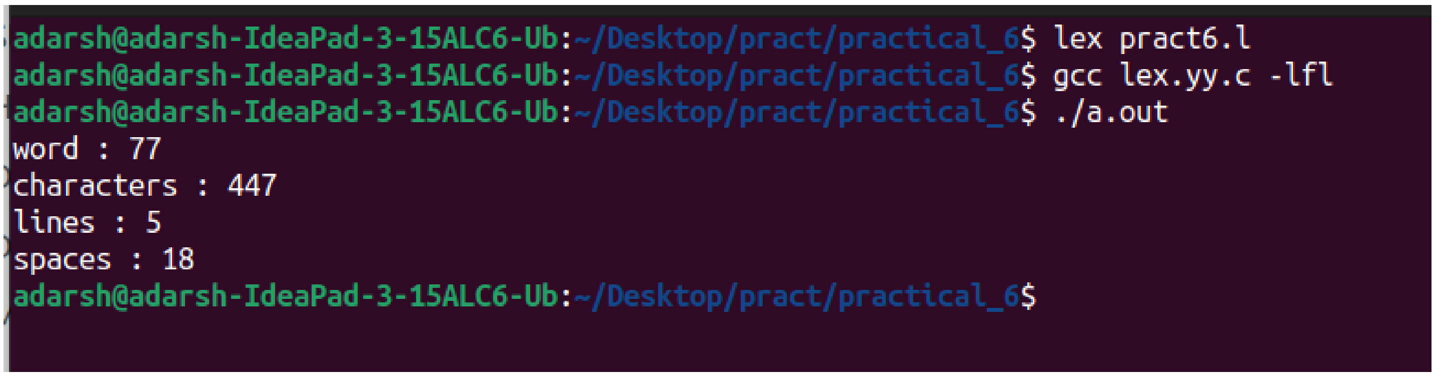
```
%{
%}
%%
[0-9]* {printf("Integer\n");}
[0-9]+\.[0-9]+ {printf("Float\n"); }
int|float|if|else|printf|main|exit|switch {printf("Keyword\n");}
[+|*|/|%|&] {printf("Operators\n");}
"-" {printf("Operators\n");}
"/*".*"*/" {printf("comment\n");}
[_a-zA-Z][_a-zA-Z0-9]{0,30} {printf("Identifier\n");}
. {printf("Invalid\n");}
%%
int main()
{
yyin=fopen("code.c","r");
yyout=fopen("kmd.txt","w");
yylex();
}
```


```
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~$ cd Desktop/
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop$ cd pract/
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract$ cd practical_4
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_4$ lex prac4.l
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_4$ gcc lex.yy.c -lfl
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_4$ ./a.out
Invalid
Identifier
Invalid
Invalid
Identifier
Invalid
Identifier
Invalid
Invalid
Invalid
Keyword
Invalid
Keyword
Invalid
Invalid
Invalid
Invalid
Invalid
Invalid
Invalid
Invalid
Invalid
Invalid
Invalid
Keyword
Invalid
Identifier
Invalid
Invalid
Identifier
Invalid
Invalid
Operators
Operators
Invalid
Identifier
Invalid
Identifier
Invalid
Identifier
Invalid
Invalid
Invalid
```

Practical 5

5. Write a Lex program to count the number of identifiers in a C file.

```
%{
#include<stdio.h>
int word=0,character=0,space=0,lines=0;
%}
%%
[A-Za-z|0-9]+ {word++;character=character+strlen(yytext);}
. {character++;}
\n {lines++;character++;}
[ \n\t\r]+ {space++;}
%%
int main(int agrc,char **argv)
{
yyin=fopen("pla.txt","r");
yylex();
printf("word : %d\n",word);
printf("characters : %d\n",character);
printf("lines : %d\n",lines);
printf("spaces : %d\n",space);
}
```

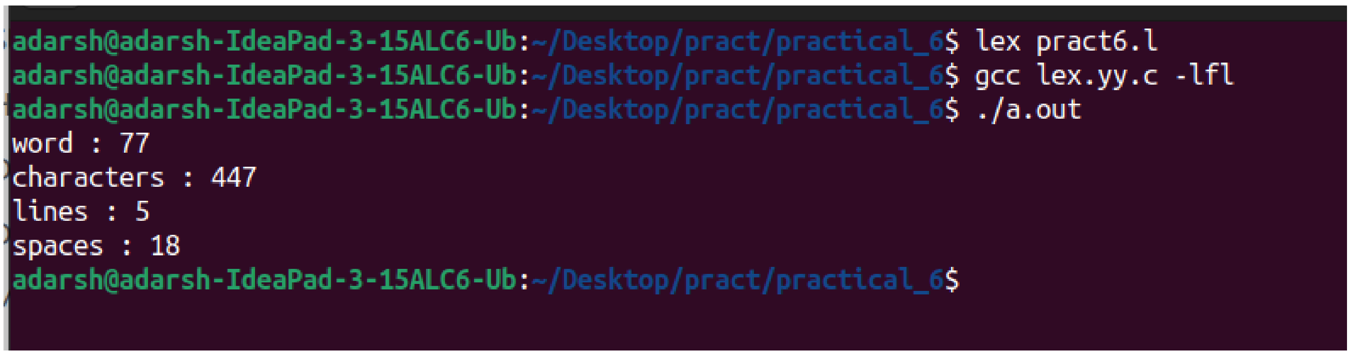


```
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_6$ lex pract6.l
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_6$ gcc lex.yy.c -lfl
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_6$ ./a.out
word : 77
characters : 447
lines : 5
spaces : 18
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_6$
```

Practical 6

6. Write a Lex program to count the number of words, characters, blank spaces and lines in a C file.

```
%{
#include<stdio.h>
int word=0,character=0,space=0,lines=0;
%}
%%
[A-Za-z|0-9]+ {word++;character=character+strlen(yytext);}
. {character++;}
\n {lines++;character++;}
[ \n\t\r]+ {space++;}
%%
int main(int agrc,char **argv)
{
yyin=fopen("pla.txt","r");
yylex();
printf("word : %d\n",word);
printf("characters : %d\n",character);
printf("lines : %d\n",lines);
printf("spaces : %d\n",space);
}
```

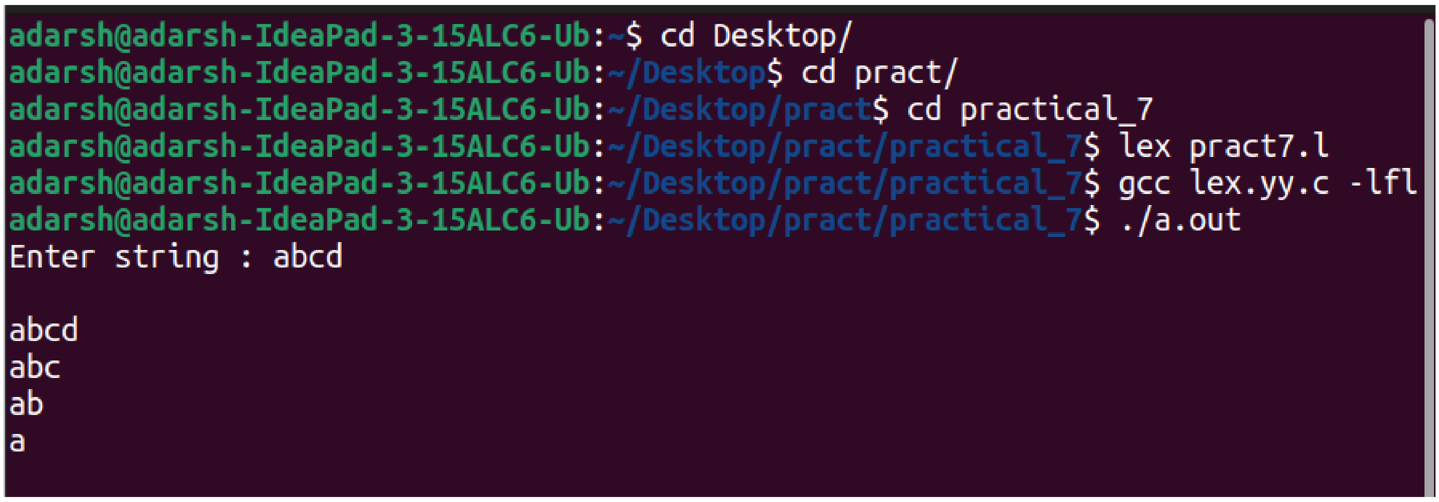


```
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_6$ lex pract6.l
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_6$ gcc lex.yy.c -lfl
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_6$ ./a.out
word : 77
characters : 447
lines : 5
spaces : 18
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_6$
```

Practical 7

7. Write a Lex specification program that generates a C program which takes a string “abcd” and prints the following output. abcd abc ab a

```
%{
%}
%%
[A-Za-z]+ {int len=yyleng;
 int i=len;
 printf("\n");
 while(i>=0)
 {
 int j=0;
 while(j<i)
 {
printf("%c",yytext[j]);
j++;
 }
 printf("\n");
 i--;
 }
 }
%%
int main()
{
printf("Enter string : ");
yylex();
}
```



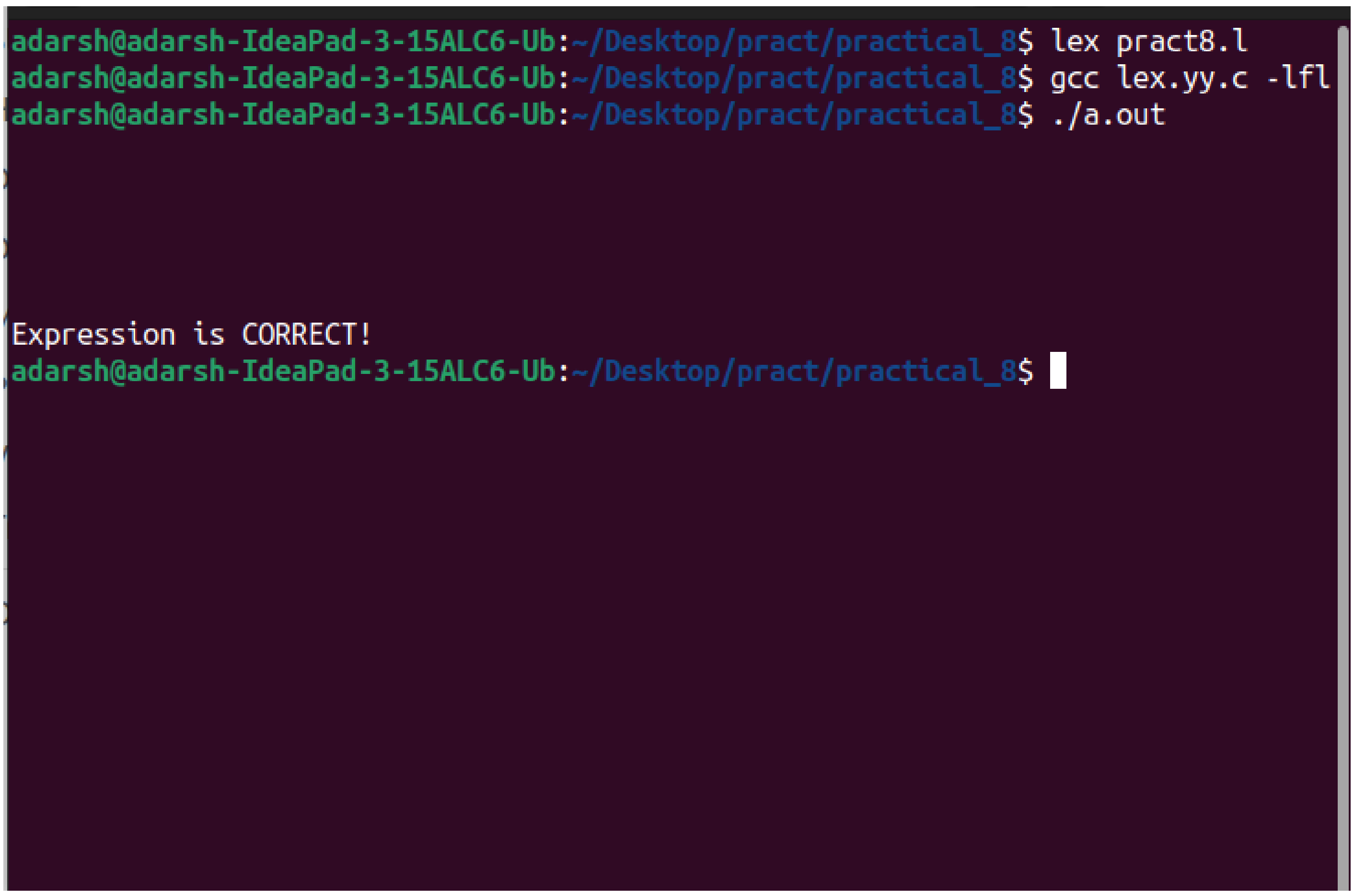
```
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~$ cd Desktop/
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop$ cd pract/
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract$ cd practical_7
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_7$ lex pract7.l
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_7$ gcc lex.yy.c -lfl
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_7$ ./a.out
Enter string : abcd

abcd
abc
ab
a
```

Practical 8

8. A program in Lex to recognize a valid arithmetic expression.

```
%{
#include<strings.h>
int opcount=0,intcount=0,check=1,top=0;
%}
%%
['('] {check=0;}
[')'] {check=1;}
[+|*|/|-] {opcount++;}
[0-9]+ {intcount++;}
. {printf("Invalid Input only digits and +|-|*|/ is valid\n");}
%%
int main()
{
yyin=fopen("abd.txt","r");
yylex();
if(intcount=opcount+1)
{
if(check==1)
{
 printf("Expression is CORRECT!\n");
}
else{
 printf("')' bracket missing from expression\n");
}
}
else{
 printf("Expression is INCORRECT!\n");
}
}
```



```
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_8$ lex pract8.l
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_8$ gcc lex.yy.c -lfl
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_8$ ./a.out





Expression is CORRECT!
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_8$ 
```

Practical 9

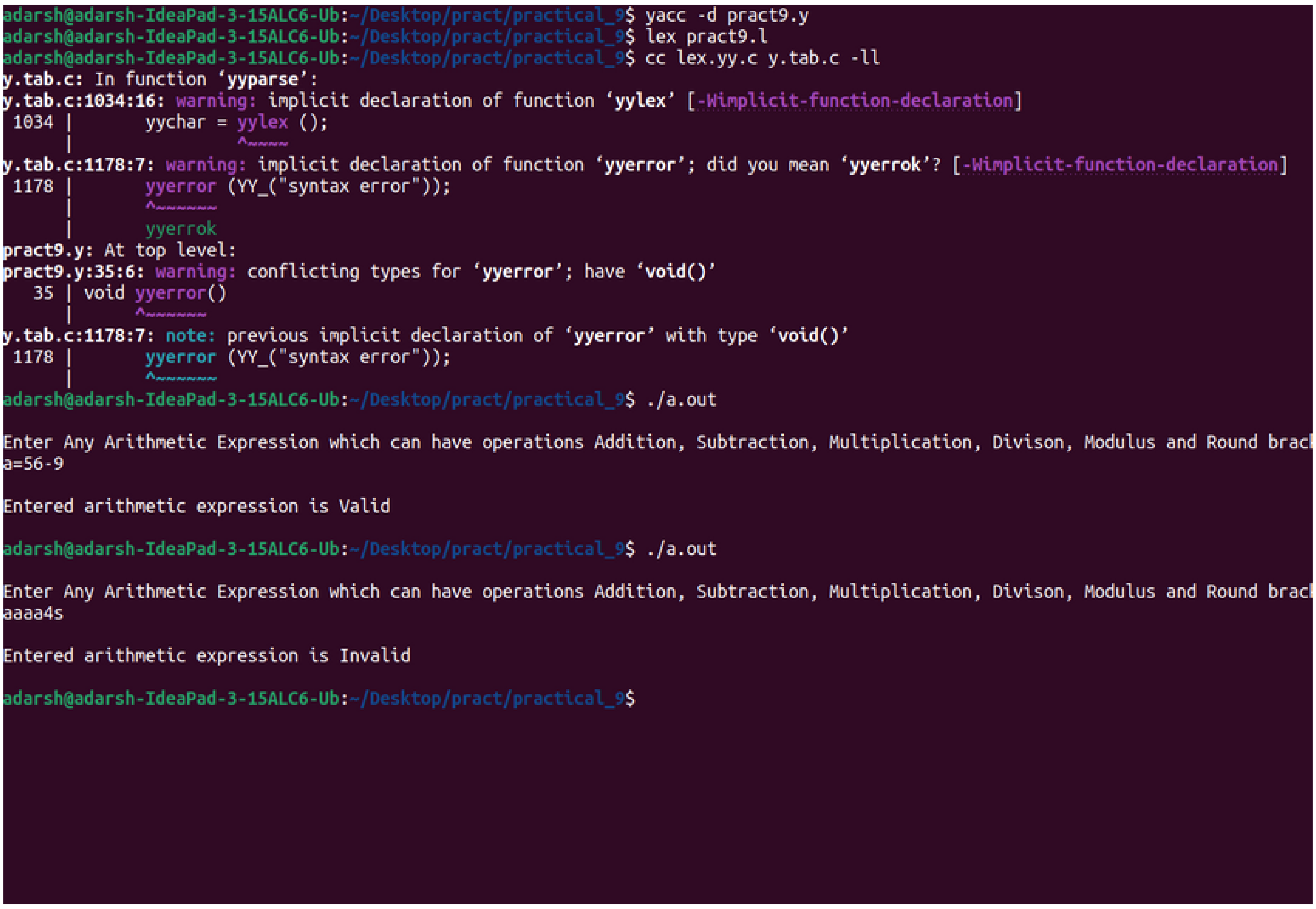
9.Write a YACC program to find the validity of a given expression (for operators + - * and /)

LEX Prog

```
%{
#include<stdio.h>
#include "y.tab.h"
%}
%%
[a-zA-Z]+ return VARIABLE;
[0-9]+ return NUMBER;
[\t] ;
[\n] return 0;
. return yytext[0];
%%
int yywrap()
{
return 1;
}
```

YACC Prog

```
%{
 #include<stdio.h>
%}
%token NUMBER
%token VARIABLE
%left '+' '-'
%left '*' '/' '%'
%left '(' ')'
%%
S: VARIABLE'='E {
 printf("\nEntered arithmetic expression is Valid\n\n");
 return 0;
 }
E:E'+'E
|E'-'E
|E'*'E
|E'/'E
|E'%'E
|'('E')'
| NUMBER
| VARIABLE
;
%%
void main()
{
 printf("\nEnter Any Arithmetic Expression which can have operations
Addition,
Subtraction, Multiplication, Divison, Modulus and Round brackets:\n");
 yyparse();
}
void yyerror()
{
 printf("\nEntered arithmetic expression is Invalid\n\n");
}
```



```
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_9$ yacc -d pract9.y
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_9$ lex pract9.l
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_9$ cc lex.yy.c y.tab.c -ll
y.tab.c: In function ‘yyparse’:
y.tab.c:1034:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-function-declaration]
 1034 |       yychar = yylex ();
      |                ^~~~~
y.tab.c:1178:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wimplicit-function-declaration]
 1178 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
      |       yyerrok
pract9.y: At top level:
pract9.y:35:6: warning: conflicting types for ‘yyerror’; have ‘void()’
   35 | void yyerror()
      |      ^~~~~~~
y.tab.c:1178:7: note: previous implicit declaration of ‘yyerror’ with type ‘void()’
 1178 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_9$ ./a.out

Enter Any Arithmetic Expression which can have operations Addition, Subtraction, Multiplication, Divison, Modulus and Round brac
a=56-9

Entered arithmetic expression is Valid

adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_9$ ./a.out

Enter Any Arithmetic Expression which can have operations Addition, Subtraction, Multiplication, Divison, Modulus and Round brac
aaaa4s

Entered arithmetic expression is Invalid

adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_9$
```

Practical 10

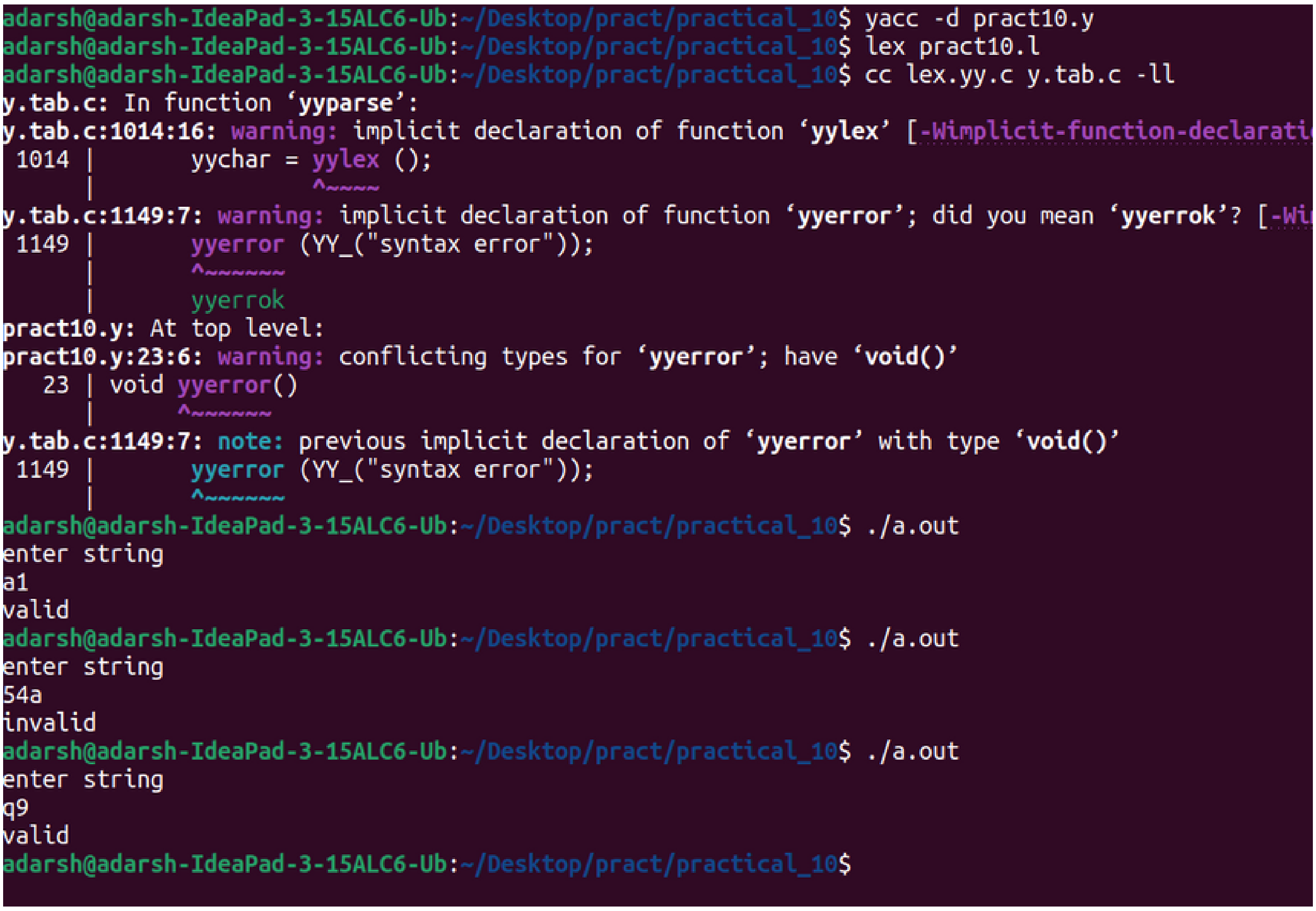
10. A Program in YACC which recognizes a valid variable which starts with letter followed by a digit. The letter should be in lowercase only.

Lex program:

```
%{
#include "y.tab.h"
%}
%%
[0-9]+ {return DIGIT;}
[a-z]+ {return LETTER;}
[ \t] {;}
\n { return 0;}
. {return yytext[0];}
%%
```

Yacc program:

```
%{
#include<stdio.h>
#include<stdlib.h>
%}
%token DIGIT LETTER
%%
stmt:A
 ;
A: LETTER B
;
B: LETTER B
| DIGIT B
| LETTER
| DIGIT
;
%%
void main(){
printf("enter string \n");
yyparse();
printf("valid \n");
exit(0);
}
void yyerror()
{
printf("invalid \n");
exit(0);
}
```



```
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_10$ yacc -d pract10.y
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_10$ lex pract10.l
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_10$ cc lex.yy.c y.tab.c -ll
y.tab.c: In function ‘yyparse’:
y.tab.c:1014:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-function-declarati
 1014 |       yychar = yylex ();
      |                ^~~~~
y.tab.c:1149:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wi
 1149 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
      |       yyerrok
pract10.y: At top level:
pract10.y:23:6: warning: conflicting types for ‘yyerror’; have ‘void()’
   23 | void yyerror()
      |      ^~~~~~~
y.tab.c:1149:7: note: previous implicit declaration of ‘yyerror’ with type ‘void()’
 1149 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_10$ ./a.out
enter string
a1
valid
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_10$ ./a.out
enter string
54a
invalid
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_10$ ./a.out
enter string
q9
valid
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_10$
```

Practical 11

11. A Program in YACC to evaluate an expression (simple calculator program for addition and subtraction, multiplication, division).

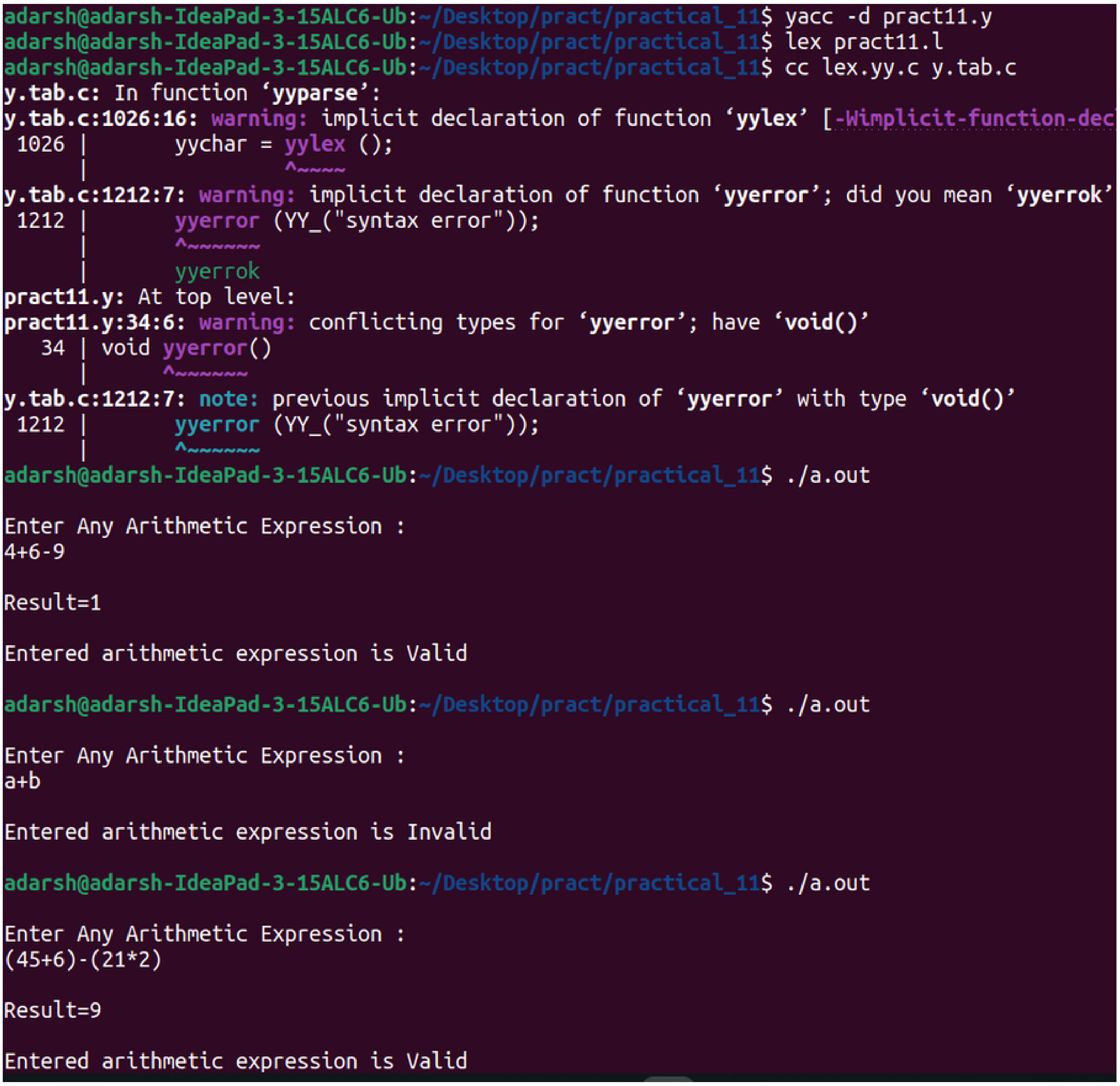
Lex Prog

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

Yacc Prog

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

%}
%token NUMBER
%left '+' '-'
%left '*' '/' '%'
%left '(' ')'
%%
ArithmeticExpression: E{
 printf("\nResult=%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("\nEnter Any Arithmetic Expression :\n");
 yyparse();
 if(flag==0)
 printf("\nEntered arithmetic expression is Valid\n\n");
}
void yyerror()
{
 printf("\nEntered arithmetic expression is Invalid\n\n");
 flag=1;
}
```



```
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_11$ yacc -d pract11.y
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_11$ lex pract11.l
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_11$ cc lex.yy.c y.tab.c
y.tab.c: In function ‘yyparse’:
y.tab.c:1026:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-function-dec
 1026 |       yychar = yylex ();
      |                ^~~~~
y.tab.c:1212:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’
 1212 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
      |       yyerrok
pract11.y: At top level:
pract11.y:34:6: warning: conflicting types for ‘yyerror’; have ‘void()’
   34 | void yyerror()
      |      ^~~~~~~
y.tab.c:1212:7: note: previous implicit declaration of ‘yyerror’ with type ‘void()’
 1212 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_11$ ./a.out

Enter Any Arithmetic Expression :
4+6-9

Result=1

Entered arithmetic expression is Valid

adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_11$ ./a.out

Enter Any Arithmetic Expression :
a+b

Entered arithmetic expression is Invalid

adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_11$ ./a.out

Enter Any Arithmetic Expression :
(45+6)-(21*2)

Result=9

Entered arithmetic expression is Valid
```

Practical 12

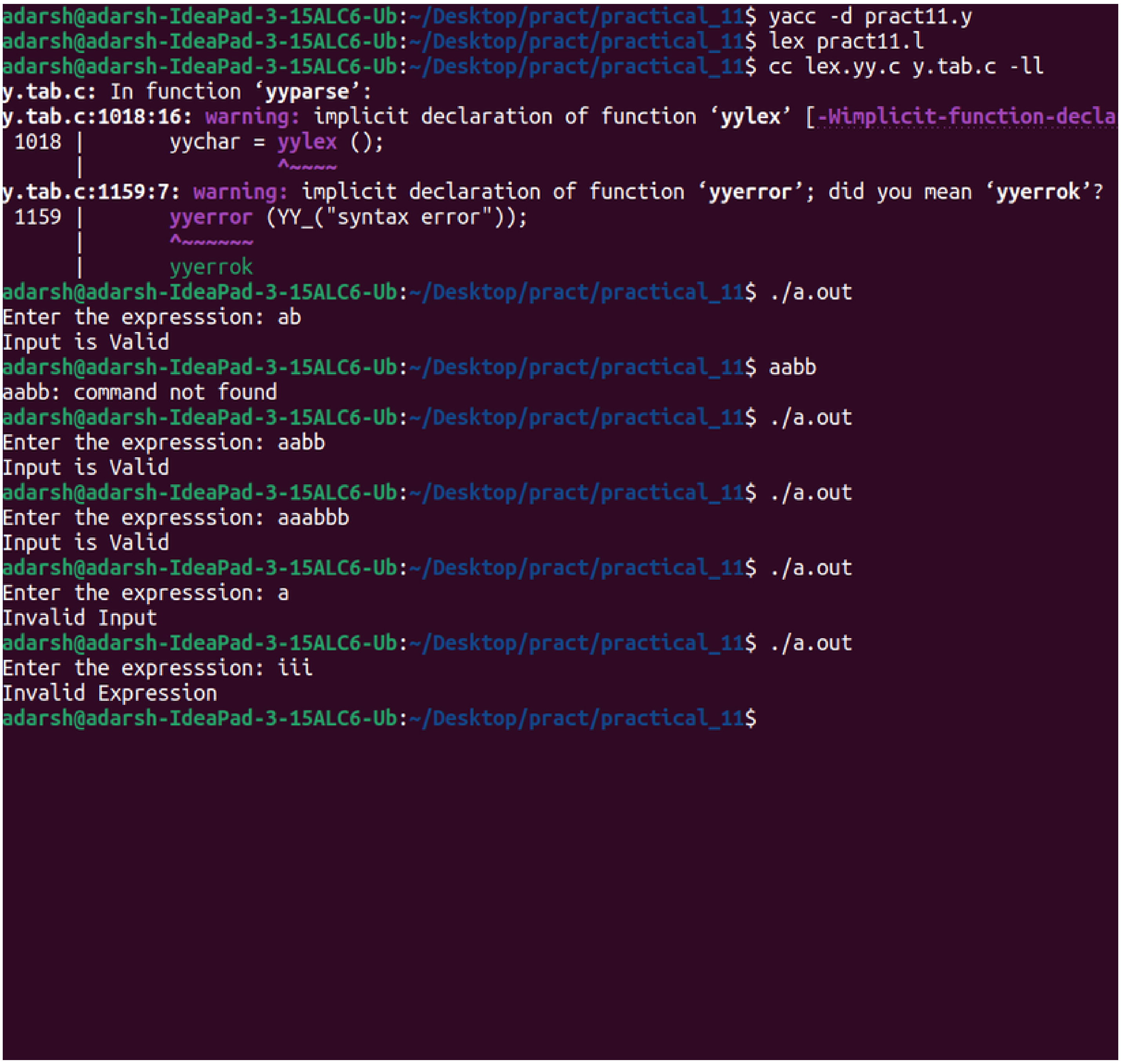
12. Program in YACC to recognize the strings “ab”, “aabb”,” aaabbb” ... of the language ($a\ n\ b\ n$, n>=1).

Lex program:

```
%{
#include "y.tab.h"
%}
alpha [Aa]
beta [Bb]
newline [\n]
%%
{alpha} { return alpha ;}
{beta} {return beta;}
{newline} { return newline ;}
. { printf("Invalid Expression\n");exit(0); }
%%
```

Yaac program:

```
%{
#include<stdio.h>
#include<stdlib.h>
#include<strings.h>
%}
%token alpha beta newline
%%
line : term newline {printf("Input is Valid\n"); exit(0);};
term: alpha term beta | ;
%%
int yyerror(char *msg)
{
printf("Invalid Input\n");
exit(0);
}
int main ()
{
printf("Enter the expresssion: ");
yyparse();
}
```



```
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_11$ yacc -d pract11.y
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_11$ lex pract11.l
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_11$ cc lex.yy.c y.tab.c -ll
y.tab.c: In function ‘yyparse’:
y.tab.c:1018:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-function-decla
 1018 |       yychar = yylex ();
      |                ^~~~~
y.tab.c:1159:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’?
 1159 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
      |       yyerrok
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_11$ ./a.out
Enter the expresssion: ab
Input is Valid
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_11$ aabb
aabb: command not found
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_11$ ./a.out
Enter the expresssion: aabb
Input is Valid
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_11$ ./a.out
Enter the expresssion: aaabbb
Input is Valid
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_11$ ./a.out
Enter the expresssion: a
Invalid Input
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_11$ ./a.out
Enter the expresssion: iii
Invalid Expression
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_11$
```

Practical 13

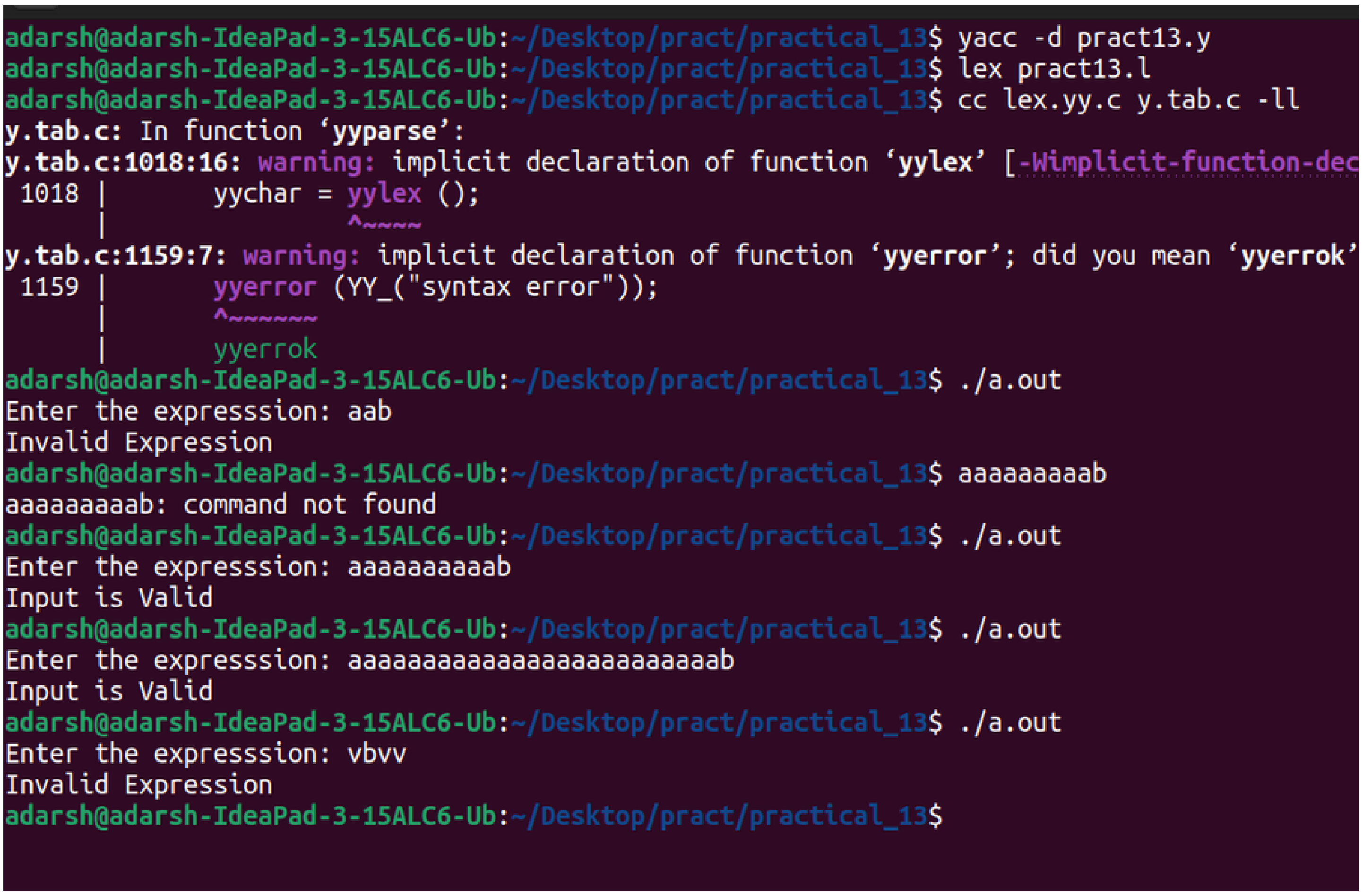
13. Program in YACC to recognize the language ($a\ nb$, n>=10). (Output to say input is valid or not)

Lex program:

```
%{
#include "y.tab.h"
%}
alpha [a]{10,}
beta [b]
newline [\n]
%%
{alpha} { return alpha ;}
{beta} {return beta;}
{newline} { return newline ;}
. { printf("Invalid Expression\n");exit(0); }
%%
```

Yaac program:

```
%{
#include<stdio.h>
#include<stdlib.h>
#include<strings.h>
%}
%token alpha beta newline
%%
line : term beta newline {printf("Input is Valid\n"); exit(0);};
term: alpha term |;
%%
int yyerror(char *msg)
{
printf("Invalid Input\n");
exit(0);
}
int main ()
{
printf("Enter the expresssion: ");
yyparse();
}
```


```
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_13$ yacc -d pract13.y
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_13$ lex pract13.l
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_13$ cc lex.yy.c y.tab.c -ll
y.tab.c: In function ‘yyparse’:
y.tab.c:1018:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-function-dec
 1018 |       yychar = yylex ();
      |                ^~~~~
y.tab.c:1159:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’
 1159 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
      |       yyerrok
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_13$ ./a.out
Enter the expresssion: aab
Invalid Expression
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_13$ aaaaaaaaab
aaaaaaaaab: command not found
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_13$ ./a.out
Enter the expresssion: aaaaaaaaaab
Input is Valid
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_13$ ./a.out
Enter the expresssion: aaaaaaaaaaaaaaaaaaaaaaaaab
Input is Valid
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_13$ ./a.out
Enter the expresssion: vbvv
Invalid Expression
adarsh@adarsh-IdeaPad-3-15ALC6-Ub:~/Desktop/pract/practical_13$
```