VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

On

COMPILER DESIGN

Submitted by

JITHENTAR A(1BM21CS082)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

Oct 2023-Feb 2024

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled **"COMPILER DESIGN"** carried out by JITHENTAR A **(1BM21CS082)**, who is bonafide student of **B. M. S. College of Engineering**. It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022-23. The Lab report has been approved as it satisfies the academic requirements in respect of Compiler Design Lab - **(22CS5PCCPD)**work prescribed for the said degree.

Sunayana S
Assistant Professor
Department of CSE
BMSCE, Bengaluru

Dr. Jyothi S Nayak
Professor and Head
Department of CSE
BMSCE, Bengaluru

1	20/11/2023	Count the number of vowels and consonants		
2	20/11/2023	Identify tokens, keywords and operators		
3	27/11/2023	Floating point numbers		
4	4/12/2023	Replacing sequence of non-empty spaces with single space		
5	11/12/2023	Recognize tokens over alphabets $0 \ldots 9$		
6.	18/12/2023	program to design lexical analyzer		
7	11/1/2024	Recursive descent		
8	11/1/2024	Desk calculator		
9	11/1/2024	String parser		
10	29/1/2024	Syntax tree generator		
11	29/1/2024	Infix to postfix using YACC		
12	29/1/2024	Three-address code generator using YACC		

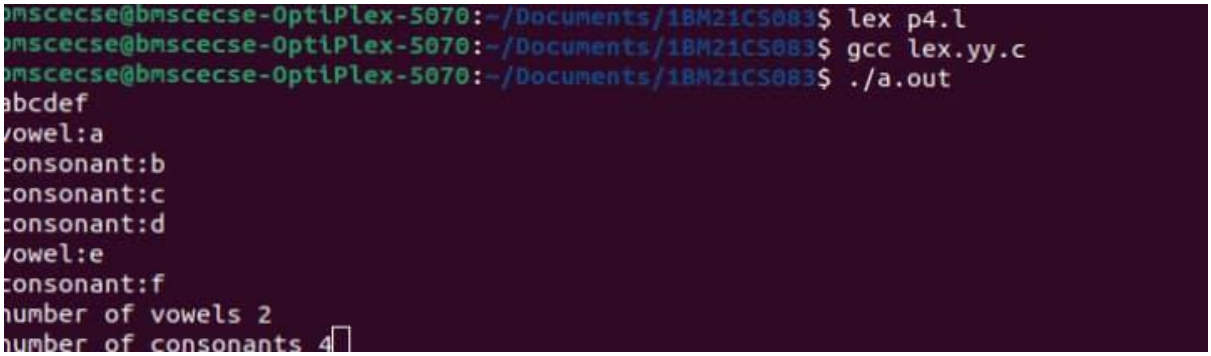
9-11-2023

1]

```
%. option noyywrap
%. {
#include <stdio.h>
%. }
%.%.
int|float|char {printf("keyword");}
[a-z A-Z] * {printf ("identifier");}
,|; {printf ("separator"); }
%.%
void main ()
{
yylex();
}
```

OUTPUT:

```
q
Id
c
Id
char
keyword
,
separator.
```


bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex p4.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ gcc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
abcdef
vowel:a
consonant:b
consonant:c
consonant:d
vowel:e
consonant:f
number of vowels 2
number of consonants 4

9-11-2023

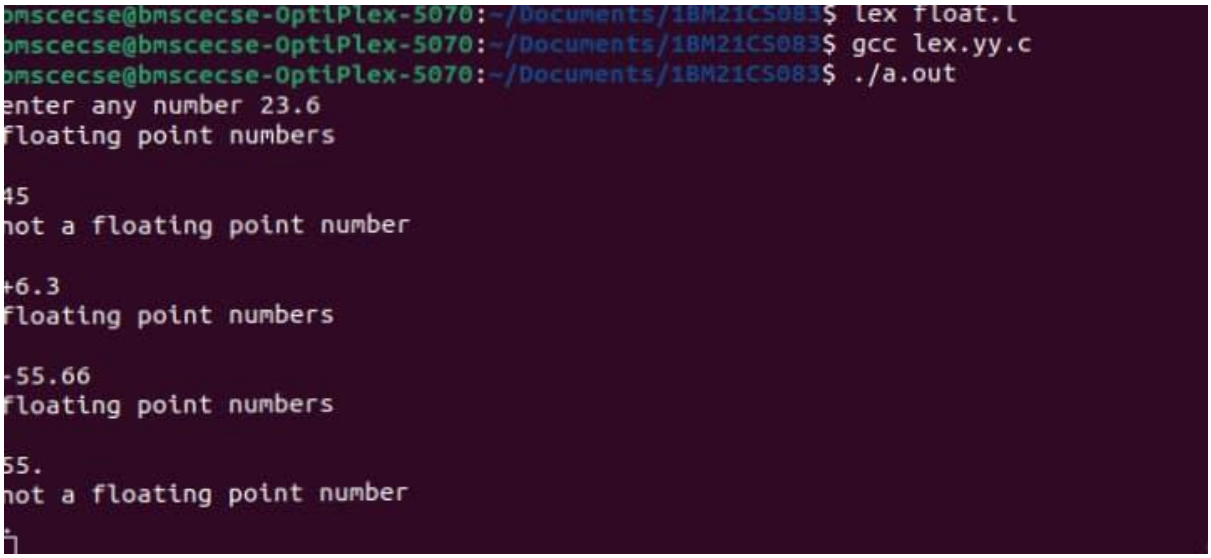
2] %.option noyywrap

```
%{
#include <stdio.h>
%}
%%
[0-9]+ { printf(" Numbers : %s\n", yytext );}
[+ -] { printf("operators : %s \n", yytext);}
[\t\n] { /* ignore whitespace and newline */}
[a-z A-Z]* { printf ( "Invalid Character : %s \n",yytext);
%%
int main()
{
printf ("Enter");
yylex ();
return 0;
}
```

OUTPUT:

```
A 2 + b 2
Invalid Character:A
Number : 2
Operator : +
Invalid Character : b
Number : 2
```

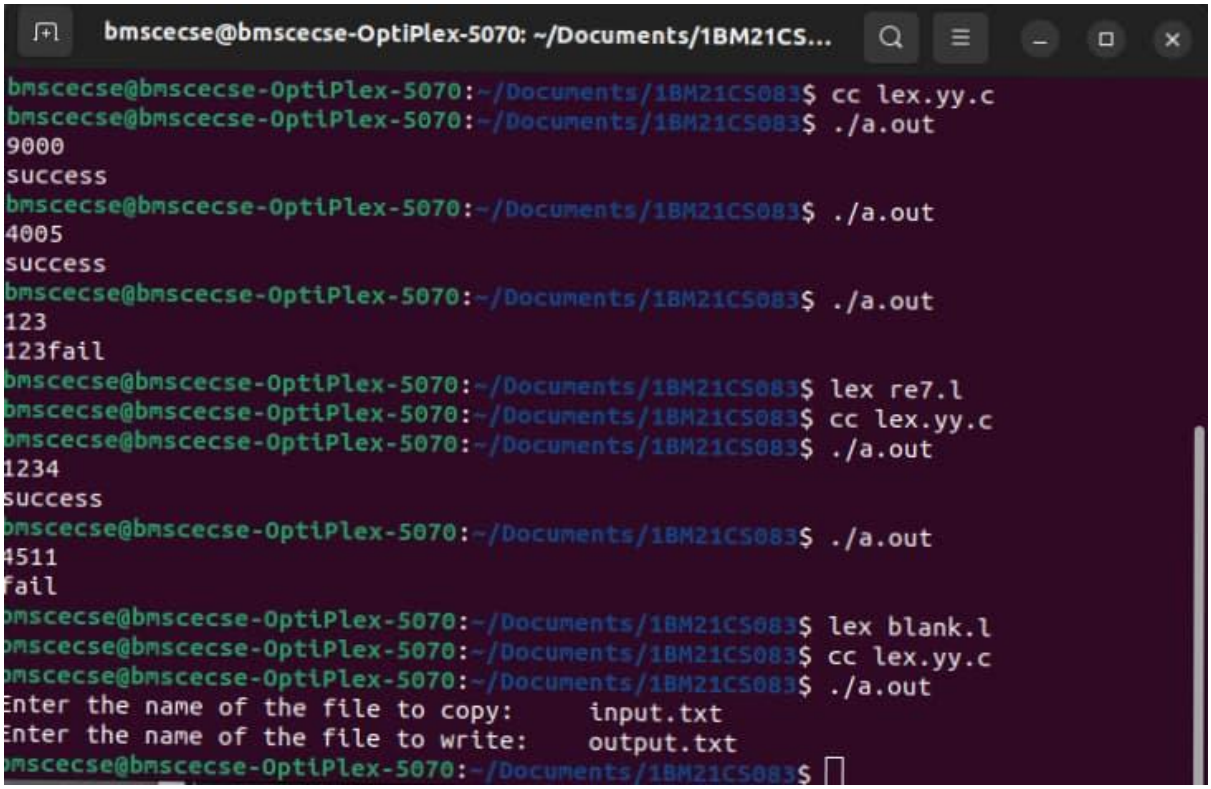
6/11/23

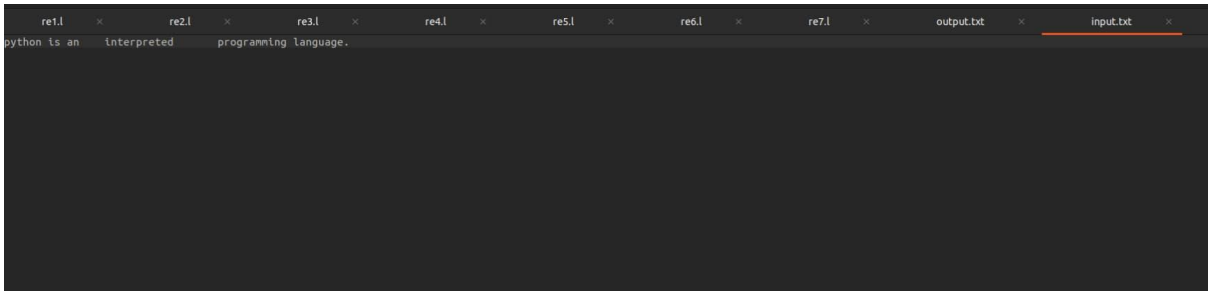

bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex float.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ gcc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
enter any number 23.6
floating point numbers
45
not a floating point number
+6.3
floating point numbers
-55.66
floating point numbers
55.
not a floating point number

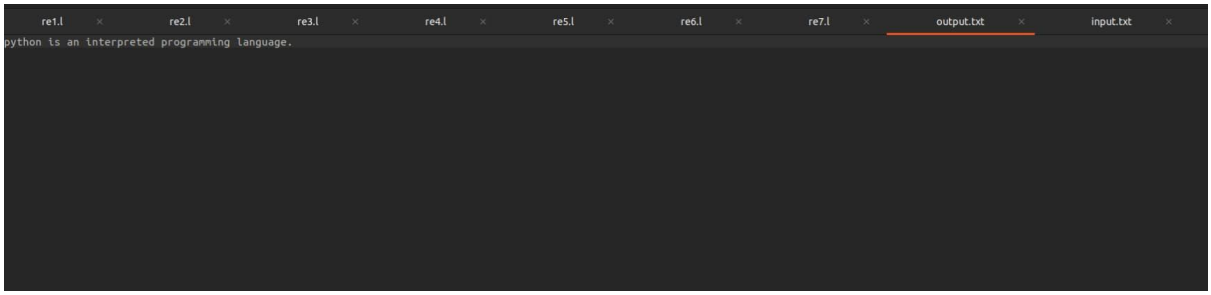
LAB-4 11-12-2023

1] Write a Lex program that copies a file replacing each non empty sequence of white spaces by a single blank.

```
%{
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
 char str1[200];
 %}
%%
[\n] { printf(yyout, "%s\n", str1); str1[0]=
 '\0'; }
[ ]* { printf(yyout, "%s", str1); str1[0] = '\0';

    strcat(str1, yytext);
<<EOF>> { printf(yyout, "%s", str1); return
%%
int main()
 {
  char filename[100];
  printf("Enter the name of the file to copy: \t");
  scanf("%s", filename);
  yyin = fopen(filename, "r");
  if (yyin == NULL)
  {
    exit();
}
```


bmscecse@bmscecse-OptiPlex-5070: ~/Documents/1BM21CS...
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
9000
success
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
4005
success
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
123
123fail
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex re7.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
1234
success
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
4511
fail
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex blank.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
Enter the name of the file to copy: input.txt
Enter the name of the file to write: output.txt
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$


re1.l
re2.l
re3.l
re4.l
re5.l
re6.l
re7.l
output.txt
input.txt
python is an interpreted programming language.


re1.l
re2.l
re3.l
re4.l
re5.l
re6.l
re7.l
output.txt
input.txt
python is an interpreted programming language.

2] Write a Lex program to recognise the following tokens over the alphabets {0, 1, ..., 9}

a) The set of all string ending in 00.

```
%option noyywrap
%{
#include<stdio.h>
%}
%%
[0-9]*00 {printf("String accepted");}
[0-9]* {printf("String not accepted");}
%%
void main(){
printf("Enter String : ");
yylex();
}
```

Output:

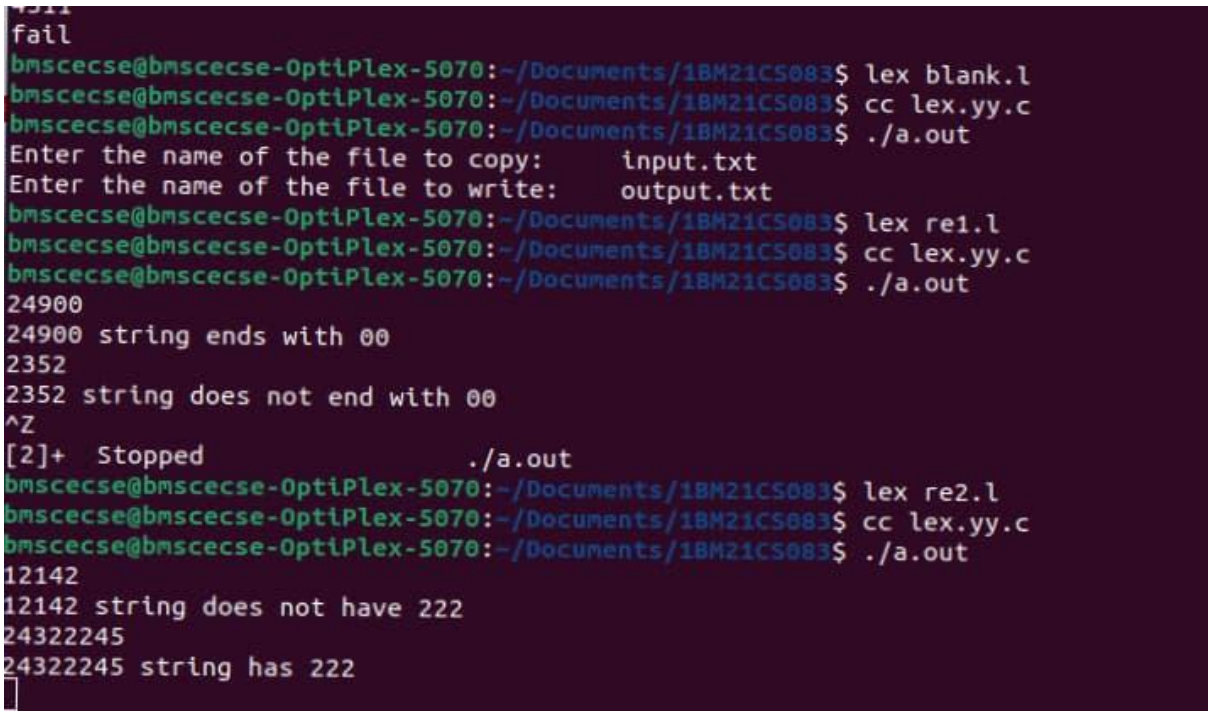
```
1100
String accepted

0011
String not accepted
```

8] The set of all 4 digits numbers whose sum is 9.

```
%%
^[0-9][0-9][0-9][0-9]$ {
  int num = atoi(yytext);
  int sum=0;
  while (num>0){
  sum+=num%10;
  num/=10;
  }
 if (sum==9){
  printf("Accepted : %s\n",yytext);
 }else{
 printf("Rejected");}

}
%%
int yywrap()
 {}
 int main(){
    yylex();
    return 0;
 }
```

8
11/12/2023


bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
1111
successbmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
11
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex re5.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
1023002245
1023002245 10th symbol from right end id 1
^Z
[1]+ Stopped ./a.out
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex re6.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
9000
success
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
4005
success
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
123
123fail


fail
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex blank.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
Enter the name of the file to copy: input.txt
Enter the name of the file to write: output.txt
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex re1.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
24900
24900 string ends with 00
2352
2352 string does not end with 00
^Z
[2]+ Stopped ./a.out
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex re2.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
12142
12142 string does not have 222
24322245
24322245 string has 222

```
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex re4.l
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
usr/bin/ld: /tmp/ccNpRHPT.o: in function `yylex':
ex.yy.c:(.text+0x33f): undefined reference to `pow'
ollect2: error: ld returned 1 exit status
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c -lm
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
01
uccessbmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c -lm
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
111
uccessbmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
1
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex re5.l
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
023002245
023002245 10th symbol from right end id 1
Z
1]+  Stopped                 ./a.out
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex re6.l
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
```

```
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex re7.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ gcc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
45612
2fail
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
1234
success
```

Write a program to design a Lexical Analyser in C/C++ /Java/Python Language.

```
def is_keyword(str):
if str == 'int' or str == 'float' or str == 'if' or
str == 'else' or str == 'while':
 return True
 else:
  return False
def is_number(str):
 n = len(str)
 k = 0
int main(){
  char input[50];
  print("Enter:");
  Scanf("%s", input);
  if (isalpha(input[0]) || input[0] == '_'){
   printf("Input is a Identifier or keyword
  }
  else if (isdigit(input[0])){
  printf("Input is a Number\n");
 }
 else{
  printf("Input is an Operator or Punctuat
 }
 return 0;
}
```

8/10

Output:

Enter a symbol or a word: int

Input is an Identifier or a keyword

18/12/2023

```
enter c code
int a = 1234 ;
Keyword: int
Identifier: a
Punctuation/Operator: =
Number: 1234
Punctuation/Operator: ;
```

8-1-2024

Write a Program to perform Recursive ~~Desc~~ Descent Parsing on the following grammar.

S -> cAd , A -> ab/a

```
#include <stdio.h>
#include <stdbool.h>

bool parse_S(char input_str[]);
bool parse_A(char input_str[]);
bool recursive_descent_Parser(char input_str[]);
int index;
bool parse_S(char input_str[]){
if (input_str[index] == 'c'){
index++;
if (parse_A(input_str) && input_str[index] == 'd')
{
index++;
return true;
} else{
return false;
}
}else{
return false;
}
}
bool parse_A(char input_str[]){
if (input_str[index] == 'a'){
int x++;
```

```
{
printf("\n Enter an arithmetic expression");
yyparse();
return 0;
}
int yyerror()
{
printf("\n Invalid expression\n");
return 0;
}
```

OUTPUT:

Enter an expression

5 + 10 + 2

Valid expression

Result: 17

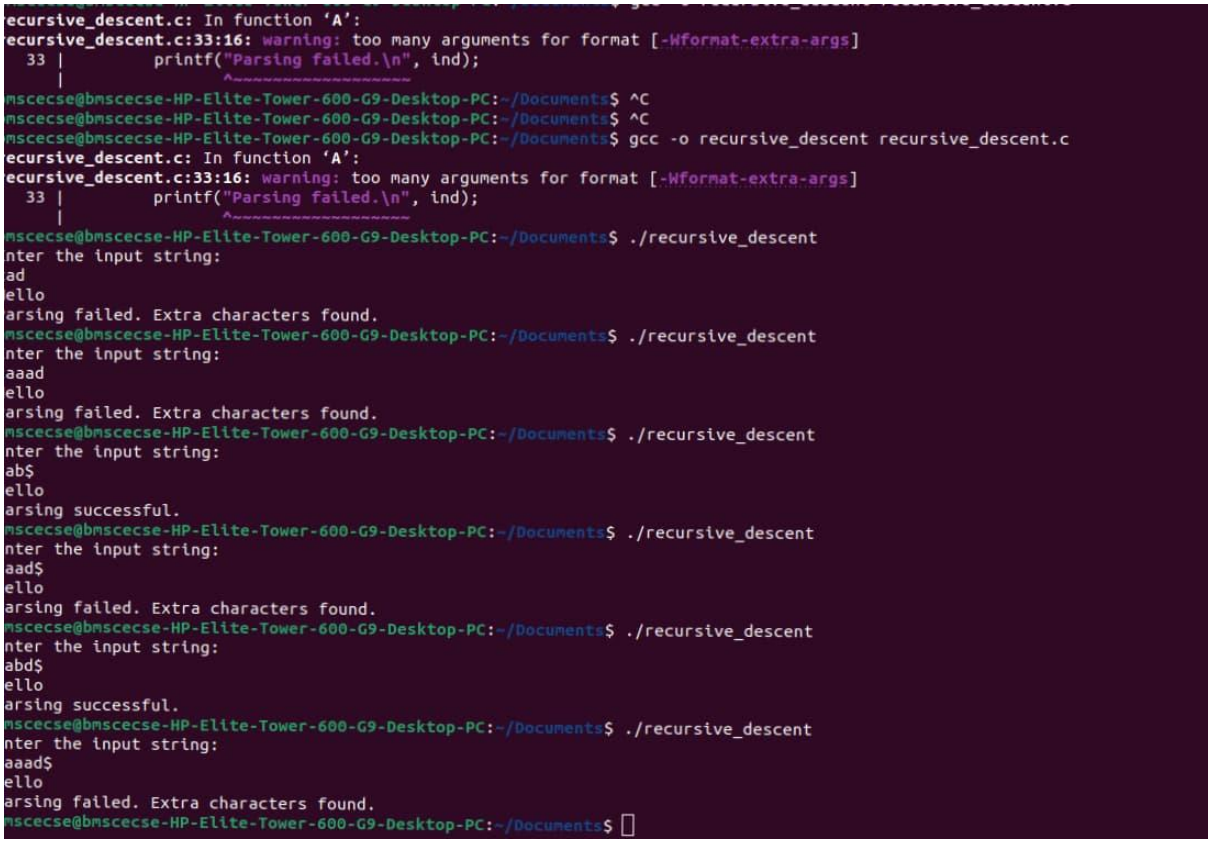
Enter an expression

10 - 5 + 2

Valid expression

Result : 7

Sun 8/1/24


ecursive_descent.c: In function ‘A’:
ecursive_descent.c:33:16: warning: too many arguments for format [-Wformat-extra-args]
33 | printf("Parsing failed.\n", ind);
| ^~~~~~~~~~~~~~~~~~~~~
mscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ^C
mscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ^C
mscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ gcc -o recursive_descent recursive_descent.c
ecursive_descent.c: In function ‘A’:
ecursive_descent.c:33:16: warning: too many arguments for format [-Wformat-extra-args]
33 | printf("Parsing failed.\n", ind);
| ^~~~~~~~~~~~~~~~~~~~~
mscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent
nter the input string:
ad
ello
arsing failed. Extra characters found.
mscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent
nter the input string:
aaad
ello
arsing failed. Extra characters found.
mscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent
nter the input string:
ab$
ello
arsing successful.
mscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent
nter the input string:
aad$
ello
arsing failed. Extra characters found.
mscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent
nter the input string:
abd$
ello
arsing successful.
mscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent
nter the input string:
aaad$
ello
arsing failed. Extra characters found.
mscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$

Write a yacc program to parse strings
grammar { $a^n b$ $n \geq 5$

anbn.l

```
%{
#include <stdio.h>
#include <stdlib.h>
#include "y.tab.h"
extern int yyval;
%}
%%
[aA]    {yylval = yytext[0]; return A;}
[bB]    {yyval = yytext[0]; return B;}
\n      {return NL;}
.       {return yytext[0];}
%%
int yywrap()
{
  return 1;
}
```

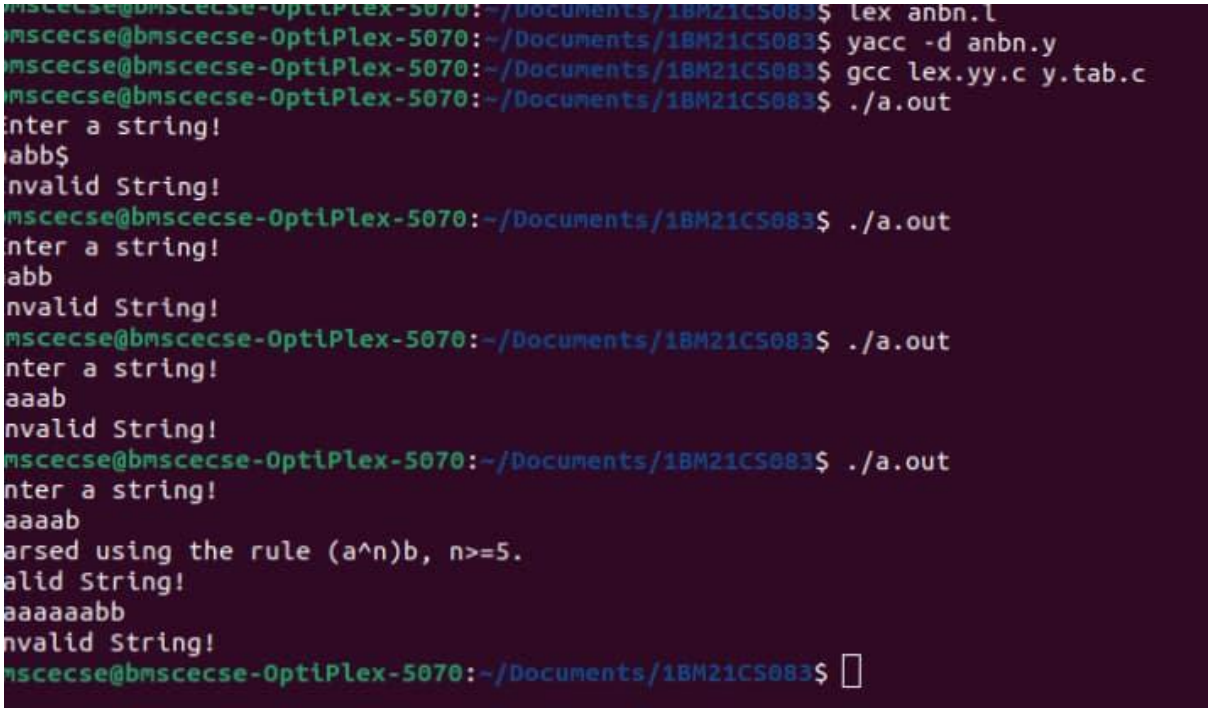
anbn.y

```
%{
#include <stdio.h>
#include <stdlib.h>
int yyerror (char *s);
int yylex (void);
%}
%token A
%token B
%token NL
%%
stmt : AAAAA SB NL {printf("Parsed using the rule (a^n b^n} n>=5, Invalid String !\n"); }
;
S: SA
|
;
%%
```

```
void main()
{
  printf ("Enter a string!\n");
  yyparse ();
}
int yyerror (char *s)
{
    printf ("Invalid string!\n");
    return 0;
}
```

Output

```
$ lex anbn.l
$ yacc -d anbn.y
$ gcc lex.yy.c y.tab.c
$ ./a.out
Enter a string!
aaaaab
Parsed using the rule (a^n)b, n>=5
Valid string!
aabb
Invalid string!
```


lex anbn.l
yacc -d anbn.y
gcc lex.yy.c y.tab.c
./a.out
nter a string!
abb$
nvalid String!
./a.out
nter a string!
abb
nvalid String!
./a.out
nter a string!
aaab
nvalid String!
./a.out
nter a string!
aaaab
arsed using the rule (a^n)b, n>=5.
alid String!
aaaaaabb
nvalid String!

8. Design a suitable grammar for evaluation of arithmetic expression having + & - operators, + has least priority and it is left associative. - has higher priority and is right associative.

```
%option noyywrap
%{
#include "y.tab.h"
%}
%%
[0-9]+ {yylval = atoi(yytext); return NUM;}
[\t];
\n return 0;
. return yytext[0];
%%
```

```
#include<stdio.h>
%
%token NUM
%left '+'
%right '-'
%%
exp: e {printf("valid expression\n"); printf("Result: %d\n", $$); return 0;}
e: e '+' e {$$ = $1 + $3;}
| e '-' e {$$ = $1 - $3;}
| NUM {$$ = $1;}
;
%%
int main()
```

```
{
printf("\n Enter an arithmetic expression");
yyparse();
return 0;
}
int yyerror()
{
printf("\n Invalid expression\n");
return 0;
}
```

OUTPUT:

Enter an expression

5 + 10 + 2

Valid expression

Result: 17

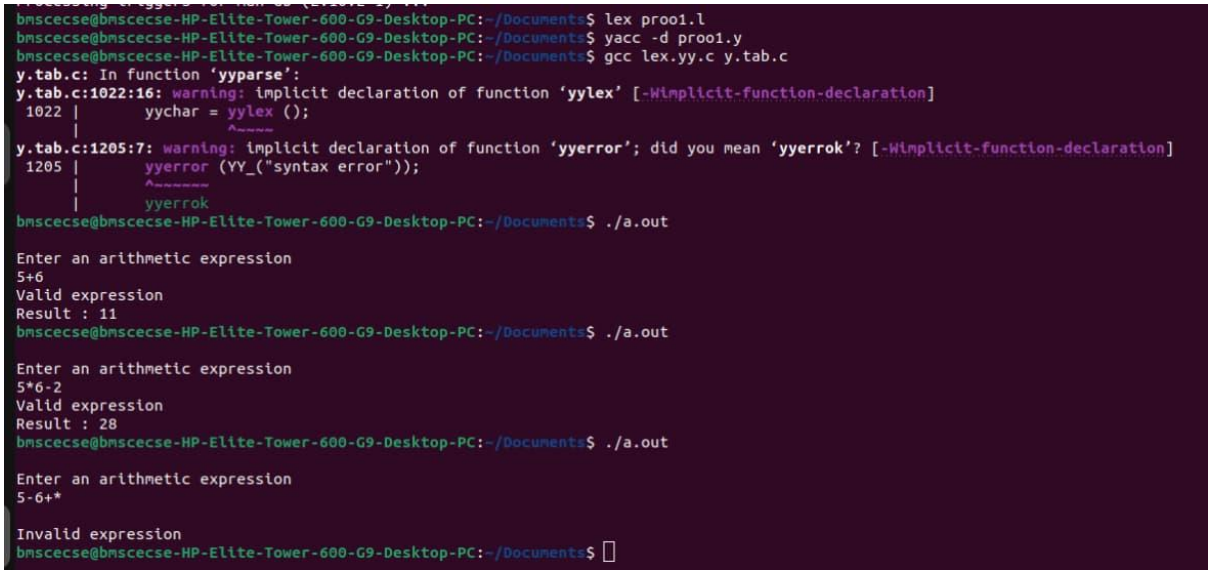
Enter an expression

10 - 5 + 2

Valid expression

Result : 7

Sun 8/1/24


bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ lex proo1.l
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ yacc -d proo1.y
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ gcc lex.yy.c y.tab.c
y.tab.c: In function ‘yyparse’:
y.tab.c:1022:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-function-declaration]
1022 | yychar = yylex ();
y.tab.c:1205:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wimplicit-function-declaration]
1205 | yyerror (YY_("syntax error"));
yyerrok
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./a.out
Enter an arithmetic expression
5+6
Valid expression
Result : 11
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./a.out
Enter an arithmetic expression
5*6-2
Valid expression
Result : 28
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./a.out
Enter an arithmetic expression
5-6+*
Invalid expression
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$

Write a YACC program to generate syntax tree for a given arithmetic expression.

p1.l

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

p1.y

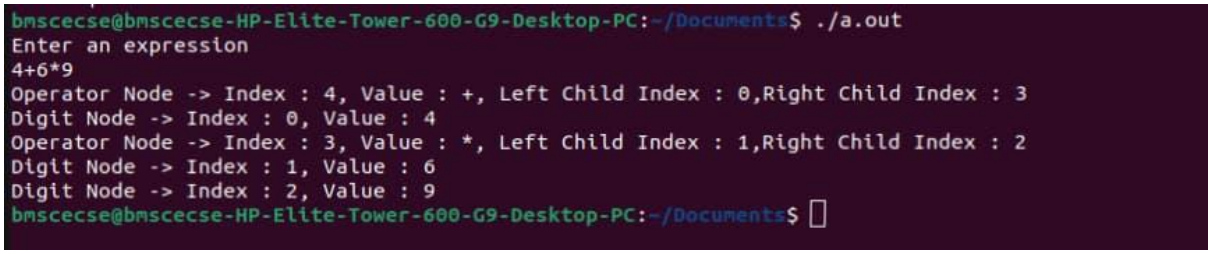
```
%{
#include <math.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct tree_node
{
char val[10];
int lc;
int rc
};
int nd;
struct tree_node syn_tree[100];
void myprint_tree(int cur_ind)
```

```
S : E{printf("\n\n");}
  ;
E : E'+'T{printf("+");}
  |T
  ;
T : T'*'F{printf("*");}
  |F
  ;
F : '('E')'
  | digit{printf("%d",$1);}
  ;
%%
int main()
{
printf("Enter infix expression: ");
yyparse();
}
yyerror()
{
printf("Error");
}
```

OUTPUT:

Enter infix Expression: 2+6*3+4

2 6 3 * + 4 +


bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./a.out
Enter an expression
4+6*9
Operator Node -> Index : 4, Value : +, Left Child Index : 0,Right Child Index : 3
Digit Node -> Index : 0, Value : 4
Operator Node -> Index : 3, Value : *, Left Child Index : 1,Right Child Index : 2
Digit Node -> Index : 1, Value : 6
Digit Node -> Index : 2, Value : 9
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$

```
%token id
%token digit
%%
S:id '=' E { printf("%s = t%d\n", iden, val_cnt-1); }
E: E '+' T { $$ = val_cnt; val_cnt++; printf("t%d =
 t%d + t%d; \n", $$, $1, $3 );
}
E: T { $$ = $1; }
;
T: T '*' F { $$ = val_cnt; val_cnt++;
        printf("t%d = t%d * t%d; \n", $$, $1, $3); }
| T '/' F { $$ = val_cnt; val_cnt++;
        printf("t%d = t%d ^ t%d; \n", $$, $1, $3); }
| P { $$ = $1; }
;
P: '(' E ')' { $$ = $2; }
| digit { $$ = val_cnt; val_cnt++; printf("t%d = %d;
  \n", $$, $1); }
;
%%
int main()
{
val_cnt = 0;
printf("Enter an expression: \n");
yyparse();
return 0;
}
yyerror()
{
```

printf("error");
}
Output:
a= 2+3*6
{ t0= 2;
t1= 3;
t2= 6;
t3=t1*t2;
t4= t0+t3;
a=t4
9-1-2024
Use YACC to convert : Infix expression to Postfix expression.

P4.l

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

```
%.{
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
%.}
%.token digit
%.%.
```

```
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex 3addcode.l
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ yacc -d 3addcode.y
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ gcc lex.yy.c y.tab.c
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
nter an expression:
=8+9-2
0 = 8;
1 = 9;
2 = t0 + t1;
3 = 2;
4 = t2 - t3;
=t4
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
nter an expression:
=2^3/23+5
0 = 2;
1 = 3;
2 = t0 ^ t1;
3 = 23;
4 = t2 / t3;
5 = 5;
6 = t4 + t5;
=t6
```

```
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex infix_to_postfix.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ yacc -d infix_to_postfix
.y
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ gcc lex.yy.c y.tab.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
Enter an infix expression:
2+4*5
245*+
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
Enter an infix expression:
3+6*2-1/3
362*+13/-
```

27-11-23

1] A Lex program in LEX to recognize floating Point No. Check for all the following input case.

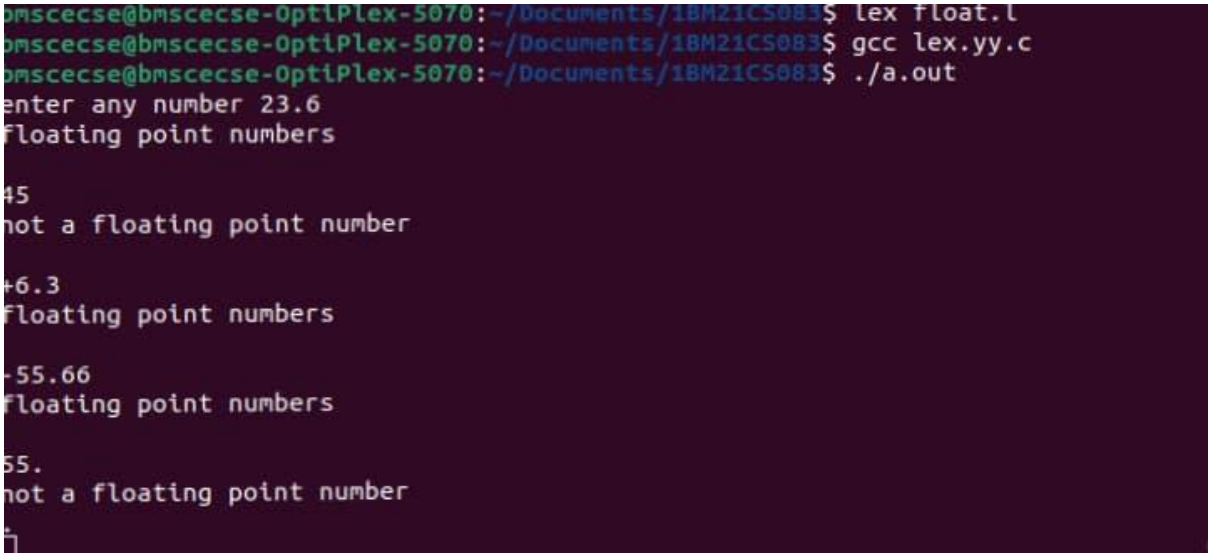
```
%.}
%.%
[+-]?[0-9]*[.][0-9]+{printf("Floating Point Number");}

[+-]?[0-9]*{printf("Not a vaild floating point Number");}

%.%
```

Output:

55
Not a valid floating point Number

33.62
Floating point Number

+13
Not a Floating point number


bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex float.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ gcc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
enter any number 23.6
floating point numbers
45
not a floating point number
+6.3
floating point numbers
-55.66
floating point numbers
55.
not a floating point number

9-11-2023

2] %.option noyywrap

```
%{
  #include <stdio.h>
 %}
%%
 [0-9]+ { printf(" Numbers : %s\n", yytext);}
 [+ -] { printf("operators : %s \n", yytext);}
 [\t\n] { /* ignore whitespace and newline */}
        }
[a-z A-Z]* { printf ( "Invalid Character : %s\n, yytext);
%%
int main()
{
 printf ("Enter");
 yylex();
 return 0;

}
```

OUTPUT:

A 2 + b 2

~~Num~~ Invalid Character: A

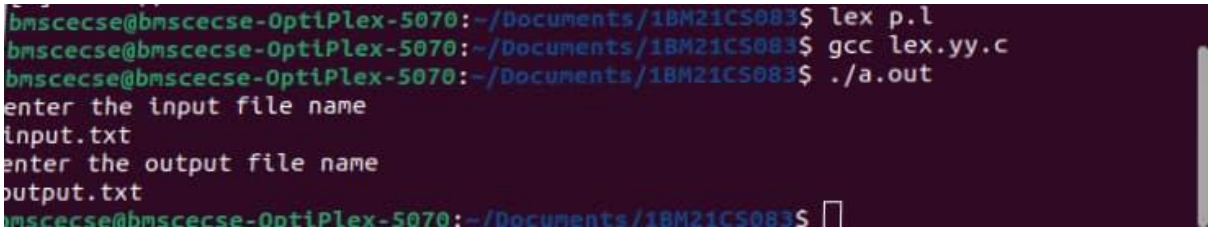
Number: 2

Operator: +

Invalid Character: b

Number: 2

6/11/23


bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex p.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ gcc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
enter the input file name
input.txt
enter the output file name
output.txt