VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
On

COMPILER DESIGN

Submitted by

HARIKA N (1BM21CS071)

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 **HARIKA N (1BM21CS071)**, 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.

Prameetha Pai
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

INDEX

Name Harika N

Standard Section Roll No.

Subject

Sl.No.	Date	Title	Page No	Teacher's Sign
		Lex	9/10	
1.	20/11/23	Count the number of vowels and consonants		
2.	20/11/23	Identify tokens, keywords and seperators		
3.	27/11/23	Floating point numbers		
4.	4/12/23	Replacing sequences of non empty spaces with single space		
5	11/12/23	Recognize tokes over alphabets		
6.	18/12/23	Program to design lexical analyzer		
7.	11/1/24	Recursive descent		
8.	11/1/24	Date calculator		

S.No.	Date	Title	Page No	Teacher's Sign
9.	11/1/24	String parser		
10	29/1/24	Syntax tree generator		
11	29/1/24	Infix to Postfix using yacc		
12	29/1/24	three address code generator using YACC		

20/11/23

Date :
Page No :

2) Write a lex program to identify each character as consonant or vowel ~~in a given sentence~~

```
%option noyywrap
%{
#include<stdio.h>
%}
%%
a|e|i|o|u|A|E|I|O|U {printf("vowel: %s\n", yytext);}
[a-zA-Z] {printf("consonant: %s\n", yytext);}
%%

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

Output

```
lex prog2.l
cc lex.yy.c
./a.out

Hari
consonant: H
vowel: a
consonant: r
vowel: i
```

```
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
```

7) Lex program to read the following input from a file and print the valid token on terminal

```
%option noyywrap
%{
#include <stdio.h>
char fname[20];
%}
%%
int|float|char {printf("keywords: %s\n", yytext);}
[0-9]* {printf("number: %s\n", yytext);}
[a-zA-Z]* {printf("character: %s\n", yytext);}
%%
```

```
void main()
{
 printf("enter the file name:");
 scanf("%s", fname);
  yyin = fopen( fname, "r");
  yylex();
  fclose(yyin);
}
```

Output

```
lex prog7.l
~~cc lex.yy.c~~ cc lex.yy.c
./a.out

enter the file name: ~~7;~~ p.txt

keywords: float
number: 0978
character: abc
```

```
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
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$
```

```
1 int a,b;
```

```
int Keywords a Identifiers, Seperatorb Identifiers; Seperator
```

27/11/23

Date :
Page No :

Lab Program

Write a program in LEX to recognize Floating point Numbers. Check for all the following input cases.

```
%{
#include<stdio.h>
%}
%%
^[+-]?[0-9]*[.][0-9]+ {printf("Floating point number");}
^[+-]?[0-9]* {printf("Not a valid floating point number");}
%%
int yywrap()
{
}
int main()
{
yylex();
return 0;
}
```

Output

```
lex lab-prg1.l
cc lex.yy.c
./a.out

99.0
Floating point number

+78.89
Floating point number

.00
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
.
```

blank.l →

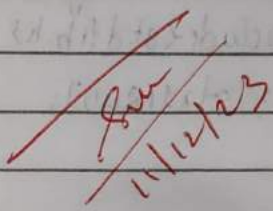
Write a lex program that copies a file, replacing each nonempty sequence of white spaces by a single blank.

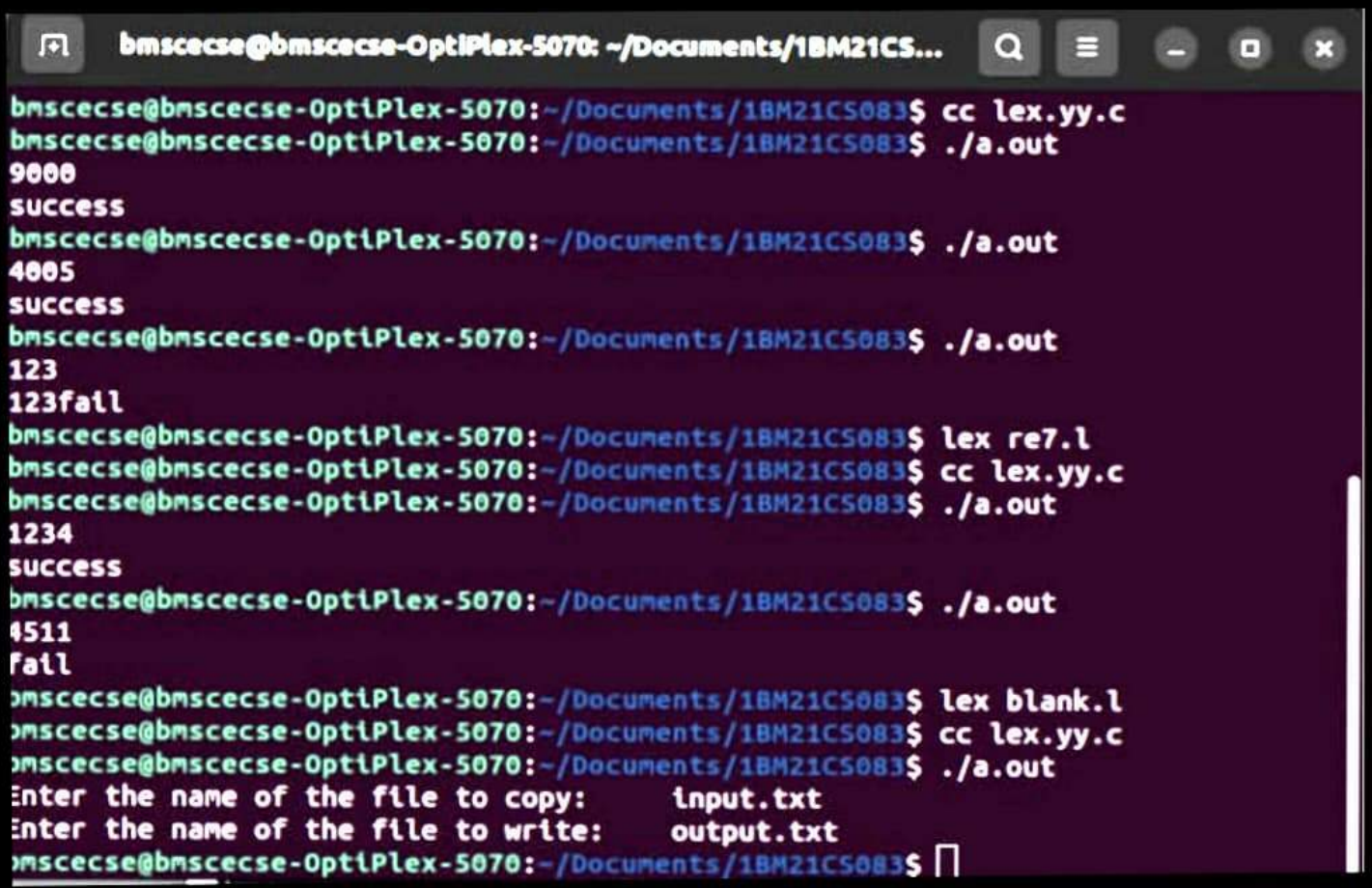
```
%{
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
char str1[200];
%}
%%
[\n]  {fprintf(yyout,"%s\n",str1); str1[0]='\0';}
[\t]
[ ]*  {fprintf(yyout,"%s",str1); str1[0]='\0';
       fprintf(yyout," %s"," ");}
      strcat(str1,yytext);
<<EOF>>  {fprintf(yyout,"%s",str1); return 0;}
%%

int main()
{  extern FILE *yyin, *yyout;
   char filename[100];
   printf("\n Enter name of file to copy: \t");
   scanf("%s",filname);
   yyin = fopen(filename,"r");
   if (yyin == NULL)
   {
      exit(0);
   }
   printf("\n Enter the name of the file to write \t");
   scanf("%s",filename);
   yyout = fopen(filename,"w");
```

```
if (yyout == NULL)
{
    exit(1);
}
yylex();

int yywrap(void)
{
}
```

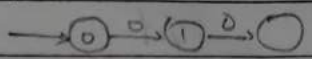



bmscecse@bmscecse-OptiPlex-5070: ~/Documents/1BM21CS...
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
9000
success
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
4005
success
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./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.

→ Write a lex program to recognize the following tokens over the alphabets {0,1,...,9}



a) the set of all string ending in 00.

```
%%
[0-9]*00 {printf("string accepted");}
[0-9]* {printf("string rejected");}
%%

int yywrap()
{
}

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

Output

```
lex string-end_00.l
cc lex.yy.c
./a.out

10100
String accepted

34560
String rejected
```

b) set of all strings with three consecutive 2's

```
%%
[0-9]* 222 [0-9]*   {printf("string accepted");}
[0-9]*              {printf("string rejected");}
```

c) set of all string such that every block of five consecutive symbols contains at least two 5's

5050

e) set of all strings such that 10th symbol from right end is 1

```
d [0-9]
%%
({d}){10}1 {d}{d}{d}{d}{d}{d}{d}{d}{d}{d} {printf("string accepted");}
%%
```

Output

0 1 2 3 4 5 6 7 8 9 1

10th symbol from right end is 1

f) set of all 4 digits whose sum is 9

```
%{
[0-9] $sum= sum + atoi(yytext); count =count+1;}
\n $if(sum%9==0 && count==4) $printf("yes\n");
                sum=0; count=0;} else
                $printf("no\n"); sum=0; count=0;$}
.$}
-/%.

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

strtok

int a;

int
a
;

Output

0090
yes

1233
Yes

1211
no

```
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
```

18/12/23

Write a program to design lexical Analyzer in C/C++/Java/Python Language (to recognize any 5 keywords, identifiers, numbers, operators & punctuations)

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

void lexicalAnalyzer(char input_code[]){
    char *keywords[] = {"if", "else", "while", "for", "return"};
    char *operators[] = {"+", "-", "*", "/", "=", "==", "<",
                         ">", "<=", ">="};
    char *punctuations[] = {",", ";", "(", ")", "{", "}"};

    char *token = strtok(input_code, " \t\n");

    while (token != NULL){
        if (isdigit(token[0])){
            printf("Number: %s\n", token);
        }
        else if (isalpha(token[0]) || token[0] == '_')
        {
            int isKeyword = 0;
            for(int i=0; i< sizeof(keywords)/sizeof(keywo
                words[0]); i++){
                if (strcmp(token, keywords[i]) == 0)
                {
                    printf("Keyword: %s\n", token);
                    isKeyword = 1;
                    break;
                }
            }
        }
```

```
        if (!isKeyword)
        {
            printf("Identifier : %s\n", token);
        }
    }

    else if (strchr("+-*/=<>;(),", token[0]) != NULL)
    {
        printf("Punctuation/Operator : %s\n", token);
    }

    token = strtok(NULL, " \t\n");
  }
}

int main() {
    char input_code[] = "if (x > 0) { return x; }
                         else { return -x; }";
    lexicalAnalyzer(inputCode);
    return 0;
}
```

Output

```
Keyword : if
Punctuation/Operator : (
Identifier : x
Punctuation/Operator : >
Number : 0
Punctuation/Operator : )
Keyword : return
Identifier : x
Punctuation/Operator : ;
```

Keyword: else
Keyword: return
Punctuation/Operator:-
Identifier: x
Punctuation/operator: ;

Sun
18/12/23

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

9/1/24

Write a program to perform recursive descent parsing on the following grammar

$S \rightarrow cAd$, $A \rightarrow ab \mid a$

```
#include <stdio.h>
#include <stdlib.h>
char input[100];
int ind = 0;
void match(char expected)
{
    if (input[ind] == expected)
    {
        ind++;
    }
}

void A();
void S()
{
    match('c');
    A();
    match('d');
}

void A()
{
    if (input[ind] == 'a')
    {
        printf("Hello\n");
        match('a');
        match('b');
    }
```

```
else
{
    printf ("Parsing failed.\n", end);
    exit(1);
}

int main() {
    printf ("Enter the input string:\n");
    scanf ("%s", input);

    S();
    if (input[ind] == '$'){
        printf ("Parsing successful.\n");
    }
    else
    {
        printf ("Parsing failed. Extra characters
                found.\n");
    }
    return 0;
}
```

Output:-

Enter a string to parse: cabd
Parsing successful. Input follows the grammar.

```
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$
```

Date :
Page No :

String match $a^n b$ where $n \geq 5$

week7_string_matching.l

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

week7_string_matching.y

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

%token A
%token B
%token NL
```

Date :
Page No :

```
%%
```

smtx: A A A A A S B B NL {printf("Parsed using the rule CanBn n>=5. \n Valid string\n");}

```
%%
S : S A
|
;

%%

void main()
{
  printf("\nEnter a string!\n");
  yyparse();
}

int yyerror(char *s)
{
  printf("Invalid String!\n");
  return 0;
}
```

Output

```
lex week7_string_matching.l
yacc -d week7_string_matching.y
gcc lex.yy.c y.tab.c
./a.out
```

Enter a string!
aaa aab

Parsed using the rule (a^n)b, n>=5.
Valid String!

aabbb
Invalid String!

```
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex anbn.l
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ yacc -d anbn.y
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ gcc lex.yy.c y.tab.c
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
nter a string!
abb$
nvalid String!
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
nter a string!
abb
nvalid String!
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
nter a string!
aaab
nvalid String!
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
nter a string!
aaaab
arsed using the rule (a^n)b, n>=5.
alid String!
aaaaaabb
nvalid String!
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$
```

→ Design a suitable grammar for evaluation of arithmetic expression having +, −, *, /, %, ^ operators

^ highest & right
% second higest & right left
*, / " " & left
+, − least & left

proo1.y

```
%{
#include<stdio.h>
%}
%token NUM
%left '+' '-'
%left '*' '/'
%left '%'
%right '^'
%%
expr: e {printf("Valid expression\n"); printf("Result: %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'^'e    {$$ = $3 ^ $1;}
 | NUM      {$$ = $1;}
;
%%
```

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

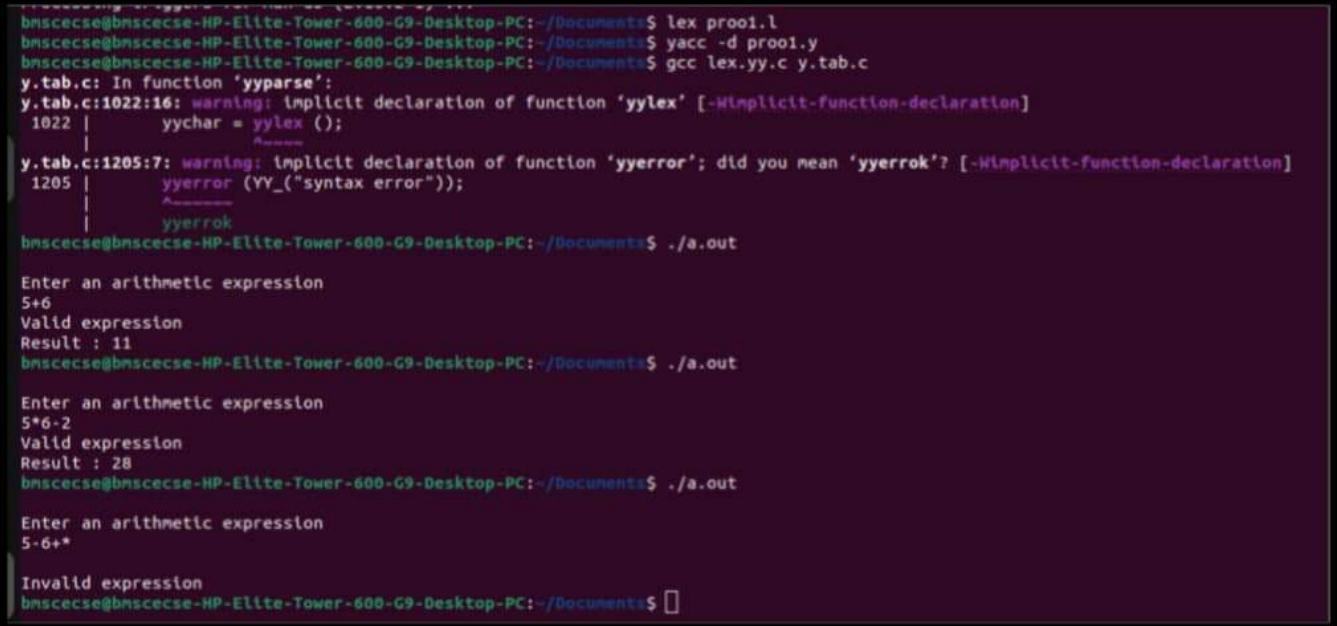
proo1.l

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

Output:-

```
lex proo1.l
yacc -d proo1.y
gcc lex.yy.c y.tab.c
./a.out
```

Enter an arithmetic expression
5 + 6
Valid expression
Result: 11


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.y

```
%{
#include <math.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct tree_node
{
 char val[10];
 int rc;
};

int end;
struct tree_node syn_tree[100];
void my_print_tree(int cur_ind);
int mknode(int lc, int rc, char val[10]);
%}

%token digit
%%
S: E { my_print_tree($1); }
;

E: E '+' T  $$ = mknode($1, $3, "+"); ; }
| T  { $$ = $1; }
;
```

```
T: T '*' F    $$ = mknode ($1, $3, "*");}
| F           $$ = $1;}
;

F: '(' E ')'  $$ = $2;}
| digit       {char buf[10]; sprintf(buf, "%d", yylval);
               $$ = mknode(-1,-1,buf);}
%%

int main()
{
  end = 0;
  printf("Enter an expression\n");
  yyparse();
  return 0;
}

int yyerror()
{
  printf("NITW Error\n");
}

int mknode(int lc, int rc, char val[10])
{
  strcpy(syn_tree[ind].val, val);
  syn_tree[ind].lc = lc;
  syn_tree[ind].rc = rc;
  end++;
  return ind-1;
}
```

```
void my_print_tree(int cur_ind)
{
  if(cur_ind == -1) return;
  if(syn_tree[cur_ind].lc == -1 && syn_tree[cur_ind].rc == -1)
  printf("Digit Node -> Index: %d, Value: %s\n",
          cur_ind, syn_tree[cur_ind].val);
  else
  printf("Operator Node -> Index: %d, Value: %s,
          Left Child Index: %d, Right Child Index: %d
          \n", cur_ind, syn_tree[cur_ind].val,
          syn_tree[cur_ind].lc, syn_tree[cur_ind].rc);

  my_print_tree(syn_tree[cur_ind].lc);
  my_print_tree(syn_tree[cur_ind].rc);
}
```

p.l

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

Output

Enter an expression

2+4

Operator Node → Index: 0, Value: +, Left Child Index: 0
Right Child Index: 1

Digit Node → Index: 0, Value: 2

Digit Node → Index: 1, Value: 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$
```

Date :
Page No :

Week8_yacc_InfixToPostfix.l

```
%{
#include<stdio.h>
#include<stdlib.h>
#include "y.tab.h"
extern int yylval;
%}

%%
[0-9]+      {yylval = atoi(yytext); return num;}
[\t];
\n          {return 0;}
.           {return yytext[0];}
%%

int yywrap()
{
}
```

Week8_yacc_InfixToPostfix.y

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

```
d }.

%token num
%left '+' '-'
%left '*' '/'
%left ')'
%left '('
%right '^'
%%

S: e {printf("\n");}
;
e: e '+' t {printf("+");}
| e '-' t {printf("-");}
| t
;
t: t '*' h {printf("*");}
| t '/' h {printf("/");}
| h
;

h: f '^' h {printf("^");}
| f
;

f: '(' e ')'
| num {printf("%d", $1);}
;

%%
void main()
{
```

```
printf("Enter an infix expression: \n");
yyparse();
}
int yyerror(const char *s)
{
  printf("Invalid infix expression!\n");
  return 0;
}
```

Output:

Enter an infix expression:
3+9*8
398*+

Enter an infix expression
3+9*8/5++
398*5/+ Invalid infix expression!

```
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/-
```

Week9_AddressCode.l

```
%{
#include<stdio.h>
#include<stdlib.h>
#include "y.tab.h"
extern int yylval;
extern char iden[20];
%}
d [0-9]+
a [a-z A-Z]+
%%
{d}     {yylval = atoi(yytext); return digit;}
{a}     {strcpy(iden, yytext); yylval=1; return id;}
[\t]    {;}
\n      return 0;
.       return yytext[0];
%%
int yywrap()
{
return 1;
}
```

Addresscode.y

```
%{
#include<math.h>
#include<ctype.h>
#include<stdio.h>
```

```
int yyerror(char *s);
int yylex(void);
int var_cnt=0;
char iden[20];
%
%token id
%token digit
%%
S: id '=' E   {printf("%s = t%d\n", iden, var_cnt-1);}
E: E '+' T    {$$ = var_cnt; var_cnt++; printf("t%d
                = t%d + t%d; \n", $$, $1, $3);}
| E '-' T     {$$ = var_cnt; "_____"; "_____"
                = _____ - _____"); }
T;
T: T '*' F    {"_____" "_____" = * _____");
| T '/' F     {" = _____" "_____" = / _____");
| F           {$$ = $1;}
;
F: P '^' F    {" _____" "_____" = ^ _____");
| P           {$$ = $1;}
;
P: '(' E ')'  {$$ = $2;}
| digit       {$$ = var_cnt; var_cnt++; printf("t%d = %d\n",
                $$, $1);}
;
%%
```

```
int main()
{
  var_cnt=0;
  printf("Enter an expression:\n");
  yyparse();
  return 0;
}

int yyerror(char *s)
{
  printf("Invalid expression!");
  return 0;
}
```

Output

Enter an expression:

a=8+9-2

t0=8;

t1=9;

t2=t0+t1;

t3=2;

t4=t2-t3;

a=t4

29/1/2024

```
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
```