VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

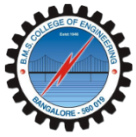
LAB REPORT
On

COMPILER DESIGN

Submitted by

KAUSHIK POTLURI (1BM21CS089)

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

B. M. S. College of Engineering,
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 **KAUSHIK POTLURI (1BM21CS089)**, 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

SURYA Gold

Date Page

PROBLEM: 10

```
%{
#include<stdio.h>
%}
%%
int|float|char    { printf("Keyword");}
[a-zA-Z]*         { printf("Identifier");}
,|;               { printf("Separator");}
%%
void main()
{
yylex();
}
```

Commands to use in terminal:

To create a file → vi filename.l

To save file → esc :wq

lex filename.l

cc lex.yy.c ← Execute file

./a.out ← generate output file.

PROGRAM : 1

Lex program to identify datatype - int, char, float

```
%option noyywrap
%{
#include<stdio.h>
%}
%%
[0-9]+      {printf("Number: %s\n", yytext);}
[+-]        {printf("Operator: %s\n", yytext);}
[\t\n]      {/* Ignore whitespace & newline */}
[a-zA-Z]    {("Invalid char: %s\n", yytext);}
%%
int main()
{
  printf("Enter");
   yylex();
   return 0;
}
```

→ OUTPUT:

Enter 04736

Number: 04736

PROGRAM : 2

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

→ OUTPUT :

Goodmorning

Goodmorning

PROGRAM: 2

Lex program to identify each character as consonant or vowel.

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

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

→ Output

kaveshik

aei = vowels

kshk = consonants

SURYA Gold

Date _____ Page _____

PROGRAM : 8

Lex program to identify alphabets as characters and numbers as digits..

```
%{
#include
%}
[0-9]* {printf("%s = digits", yytext);}
[a-zA-Z]* {printf("%s = characters", yytext);}
```

→ Output

```
15 abdc 321
15321 = digits
abdc = characters
```

PROGRAM : 4

Lex program to count number of words in an input sentence

```
%%
[a-zA-Z0-9]+    {c++;}
\n {printf("the count is %d", c);}
%%
int yywrap() {}
int main()
{
 printf ("enter the sentence");
 yylex();
 return 0;
}
```

-> Output:

```
Enter sentence: I'm in BMSCE

Count is : 3
```

PROGRAM: 5

Lex program to count no: of vowels and consonants in a string.

```
%{
    int vow_count = 0;
    int const_count = 0;
%}
%%
[aeiouAEIOU] {vow_count++;}
[a-zA-Z] {const_count++;}
\n {printf("vow_count: %d, const_count: %d", vow_count,
        const_count);}
%%
```

→ Output:

Enter: Kaushik

vow_count: 3, const_count = 4

```
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
```

PROGRAM: 6

Lex program to print invalid string if an alpha-numeric string is entered as input using regular definition

```
%%
[0-9]*      { printf ("Valid String"); }
[a-zA-Z]*   { printf ("Valid Stoing"); }
[0-9a-zA-Z]* { printf ("Invalid String"); }
%%
```

-> Output:

acbd152

Invalid string

abc

Valid String

123

Valid String

PROGRAM: 7

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

```
%%
int|float|char { printf("%s = keywords", yytext);
[0-9]*         { printf("%s = numbers", yytext);
[a-zA-Z]*      { printf("%s = characters", yytext);
%%
void main()
{ printf ("enter f-name");
  scanf("%s", fname);
  yyin = fopen(fname, "r");
   yylex();
   fclose(yyin);
```

→ Output

Enter f-name: k.txt

keywords: float, number: 0123, Character: kp

PROGRAM: 8

Modify program 7 such that the output is available is an output file. Use the same sample input.

```
%%
        {fprintf(yyout, "keywords: %s", yytext);}
        "          "numbers: %s"
        "          "characters: %s"

void main()
{
 printf("Enter input file");
 scanf("%s", fname);
 printf("Enter output file");
 scanf("%s", foname);
  yyin = fopen(fname, "r");
```

```
yyout = fopen(toname, "w");
yylex();
fclose(yyin);
fclose(yyout);
```

→ Output

Enter input file
k.txt

Enter output file
kout.txt

20/11/23

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

SURYA Gold
Date Page

PROGRAM:9

LEX program to recognize floating point numbers.

```
%%
^[+-]?[0-9]*[.][0-9]+{ printf("Floating point number"); }
^[+-]?[0-9]* { printf("Not a valid floading point number"); }
%%
```

→ Output:

5.36'
floating point number

5
Not a valid floating point number.

PROGRAM:10

Read an input sentence and check if it is compound or simple. If a sentence has the word 'and', 'or', 'but', 'because', 'if', 'then', 'nevertheless', then it is compound else it is simple.

```
%%
and|or|but|because|if|then|nevertheless { flag=1; }
[a-zA-Z]* {}
%%
int main() { return 0; }
```

→ Output:

Enter a Sentence : My Name is Kaushik
Simple Sentence

Enter a Sentence : Today is tuesday and it is cloudy
Compound Sentence

```
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
```

PROGRAM: 11

Lex to read sentence and to check if the sentence begins with English Articles (A, a, AN, An, THE and The).

```
%{
^A|a|AN|An|THE|The {flag=1;}
[a-zA-Z]* {}
.;
\n {return 0;}
%%
```

Output

Enter a sentence The tower is high

Starting with an article

Enter a sentence tower is high

Does not start with an article.

PROGRAM: 12

Lex program to check if the input sentence ends with any of the following punctuation (?, ., !)

```
%{
[a-zA-Z]* [?|.|!] {printf("ends with punctuation")}
[a-zA-Z]* {printf("does not end with punctuation")}
%%
```

Output:

Enter a sentence: The tower is high!

ends with punctuation

Enter a sentence: tower is high

does not end with punctuation.

SURYA Gold
Date Page

PROGRAM :13

Lex to check if the entered number is signed or unsigned using appropriate meta character.

```
%%
^[+-][0-9]+    {a=1;}
[0-9]+ { }
%%
```

Output:

Enter : +100

Signed

Enter: ~~100~~

Unsigned.

27/11/23

PROGRAM: 14

Lex program to count the number of comment lines (multi line comments or single line) in a program. Read the input from a file called input.txt & print the count in a file called output.txt

```
%x CMNT
%%
"/*" {BEGIN CMNT;}
<CMNT>.;
<CMNT>"*/" {BEGIN 0; CC++;}
%%
int yywrap() {}
int main (int argc, char *argv[])
{
  if(argc!=3)
  {
    printf ("Usage: %s <src_file><dest_file>\n",
        argv[0]);
    return 0;
  }
yyin = fopen (argv[1], "r");
yyout= fopen (argv[2], "w");
yylex();
printf ("\n Number of Multiline comments = %d \n", cc);
return 0;
}
```

PROGRAM : 15

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

```
%%
[\n] { printf (yyout, "%s\n", Str1); Str1[0] = '\0'; }
[ ]* { printf (yyout, "%s", Str1); Str1[0] = '\0';
       fprintf (yyout "%s", " "); }
       Str (at (str1, yytext);
<<EOF>> { printf(yyout, "%s", Str1); return 0; }
%%
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(0);
  }
```

```
printf("Enter the name of the file to write it:");
scanf("%s", filename);
yyout = fopen(filename, "w");
if (yyout == NULL)
{
  exit(1);
}
yylex();
}
int yywrap(void)
{
}
```

PROGRAM 16:

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 is ending in 00.");}
[0-9]* { printf("String is not ending in 00");}
%%
```

b) The set of all Strings with three consecutive 222's

```
%%
[0-9]*222[0-9]* { printf("String contain 222");}
[0-9]* { printf("String does not contain 222");}
%%
```

SURYA Gold
Date Page

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

```
%%
[0-9]* {
 int i, c=0;
if(yyleng<5)
{ printf("%s doesnt match any rule\n", yytext);
}
else
{
for (i=0; i<5; i++)
{ if (yytext[i]=='5')
{
 c++;
{
}
if(c>=2)
{
 for(; i<yyleng; i++)
{
if(yytext[i-5]=='5') { c--;}
if (yytext[i]=='5') { c++;
}
if (c<2) { printf("%s doesnt match any rule\n",yytext);
 break;
}
}
else
{
 printf("%s doesnt match any rule\n", yytext);
 }
 }
 }
%%
```

d) The set of all strings beginning with a 1 which, interpreted as the binary representation of an integer, is congruent to zero modulo 5.

```
%%
(1(0)*(11|01)(01*01|00*10(0)*(11|1))*0)
(1|10(0)*(11|01)(01*01|00*10(0)*(11|1))*10)*
printf("The set of all strings beggiinning with a 1 is interpreted as the binary representation of an integer.");
}
%%
```

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

```
%%
({d})*1{d}{9}{
printf("The set of all strings such that the 10th symbol from the right end is 1");
}
%%
```

f) The set of all four digits numbers whose sum is 9

```
%%
{d}{4}
int sum=0,i;
for(i=0;i<4;i++)
{
sum=sum+yytext[i]-48;
}
if(sum==9)
{
```

```
printf(" The sum of digits whose sum is 9");
}
}
%%
```

g) The set of all four digital numbers, whose individual digits are in ascending order from left to right.

```
%%
{d}{4}{
for (i=0; i<3; i++)
{
if (yytext[i] > yytext[i+1])
{
 sum=0;
 break;
}
}
if (sum==1)
{ printf(" ascending order from left to right ");
}
else { printf(" %s doesnt match any rule\n", yytext); }
}
%%
```

⇒ Output:

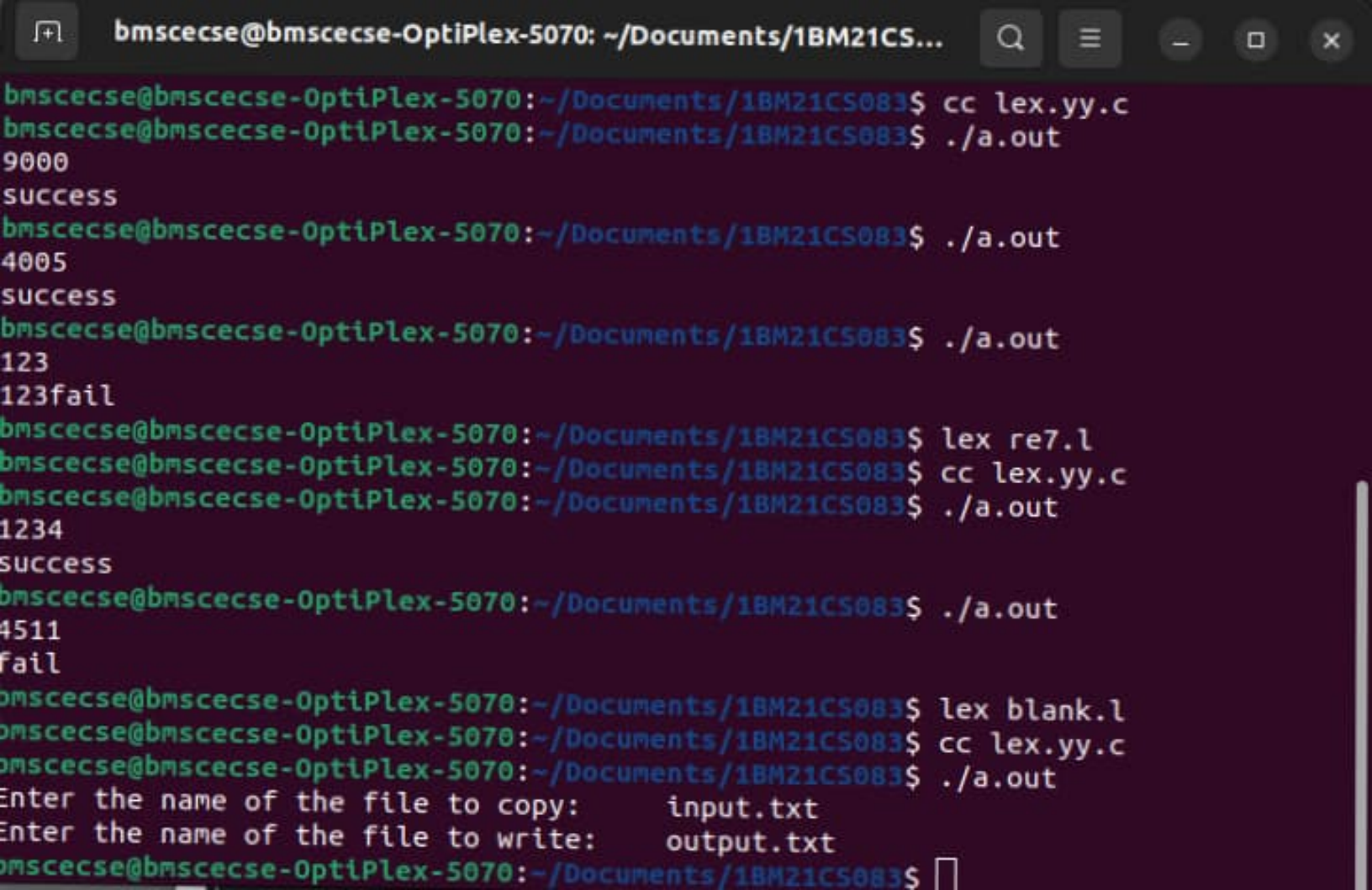
a) 10100
String accepted

34560
String rejected.

b) 1222
string accepted

12
string rejected.

4/12/2023


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$

```
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
```

```
4311
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
```

```
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
```

PROGRAM:17

Write a program to design lexical Analyser in c language to recognize any five keywords, identifiers, numbers, operators & punctuations.

```
#include<stdio.h>
#include<ctype.h>
#include<string.h>
#define KEYWORD 1
#define IDEN 2
#define NUM 3
#define OPERA 4
#define PUNCT 5
#define INVALID 6

int iskeyword(char* str)
{
  char keywords[5][10]={"if","else","while","int",
    "return"};
  for(int i=0;i<5;++i)
  {
    if(strcmp(str,keywords[i])==0)
    {
      return KEYWORD;
    }
  }
  return 0;
}

void analyzeToken(char* token)
{
  if(iskeyword(token)==KEYWORD)
  {
    printf("Keyword: %s\n",token);
  }
```

```
else if (isdigit(token[0]))
  {
    printf("Number: %s\n", token);
  }
elseif(isalpha(token[0]) || token[0] == '_')
  {
    printf("Identifier: %s\n", token);
  }
else if (strchr("+-*/=", token[0]) != NULL)
  {
  printf("Operator: %s\n", token);
  }
elseif (strchr("();,{}@!?.", token[0]) != NULL)
  {
    printf("Punctuation: %s\n", token);
  }
  else
  {
    printf("Invalid:");
  }
}
int main()
{
  char[500]; char input[500];
  Enter token[100 char token[100];
    int tokenIndex = 0;
  printf("Enter C program code:\n");
  fgets(input, sizeof(input), stdin);
  for(int i=0; i<= strlen(input); ++i)
  {
    if(isalnum(input[i]) || input[i] == '_')
    {
      token[tokenIndex++] = input[i];
    }.
```

```
else
{
if(tokenIndex > 0)
 {
  token[tokenIndex] = "10";
  analyzetoken (token);
  tokenIndex = 0;
 }
 if ( input[i] != ' ' && input[i] != '\n'
 && input[i] != '\t')
 {
   token[0] = input [i];
   token [1] = '\0';
    analyseToken (token);
 }
}
}
  return 0;
}.
```

=> Output:

```
Sum = a +b *10 ?.
Identifier : Sum
Operator : =
Identifier : a
Operator : +
Identifier : b
Operator : *
Number : 10
Punctuation : ?
Punctuation : .
```

10/10

18/12/2023

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

SURYA Gold
Date Page

PROGRAM: 18 - Recursive Decent

```
#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", ind);
    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:

(1) Enter the input string:
cad$
Hello
Parsing successful.

(2) Enter the input string:
caaad$
Hello
Parsing failed.

```
recursive_descent.c: In function ‘A’:
recursive_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
recursive_descent.c: In function ‘A’:
recursive_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$
```

SURYA Gold
Date Page

PROGRAM 19: Desk calculator.

```
%{
#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 '-'
%%
expr: e { printf("valid expression\n");
  printf("Result: %d\n", $$); return 0; }
e: e '+' e { $$ = $1 + $3; }
| e '-' e { $$ = $1 - $3; }
| NUM { $$ = $1; }
;
%%
```

→ Output:

Enter an arithmetic expression

5+6*8-1

10

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$
```

SURYA Gold
Date Page

PROGRAM: 20

→ 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 ind;
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 { $$ = mknode ($1, $3, "+"); ;}
IF {$$ = $1;}
;

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

int main()
{
 ind=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;
    ind++;
   return ind-1;
}
```

SURYA Gold
Date Page

```
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 : %d\n",
          cur-ind, syn-tree [cur-ind].val);

  else
printf("Operator Node → Index %d, Value : %c,
   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]);

  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: 2, 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$
```

SURYA Gold

Date _____ Page _____

PROGRAM: 21

→ String match $a^n b$ where $n > 5$

String.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;
}
```

String.y

```
%{
#include <Stdio.h>
#include <Stdlib.h>
int yyerror(char *s);
int yylex(void);
%}
%token A
%token B
%token NL
```

1.1.

Smtr: AAAAASBB NL { printf (" Passed using the rule $(a^n b) b$,
$n >= 5$. \n Valid String \n"); };

S: S A
|
;

1.1.

```
void main()
{
printf(" Enter a String! \n");
 yyparse();
}

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

=> Output

Enter a String!
aaaaab

parsed using the rule $(a^n|b, n>=5)$.
valid String

aabbb
Invalid String

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

SURYA Gold
Date Page

PROGRAM: 22

Infix to Postfix:

```
%{
#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()
{
}
```

Infix to Postfix.y

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

%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/-
```

PROGRAM: 23

Address Code.l

```
%}
d [0-9]+
a [a-zA-Z]+
%%
{d} {yylval = atoi (yytext); return digit;}
{a} {strcpy (iden, yytext); yylval = 1; return id;}
[\t] {;}
\n return 0;
. return yytext[0];
%%
```

Addressod.y

```
%%
S: id '=' E {printf("%s=t%d\n", iden, var_cnt-1);}
E: E '+' T {$$ = var_cnt; var_cnt++; printf("t%d
 |E '-' T          = t%. ___          ___
        ___                 ___
 ;
T: T '*' F {" ___ " ___ " ___ " ___ "};
 |T '/' F {" ___ ___ ___
 |F   {$$=$1;}
 ;
%%
```

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
```