

**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

## PROBLEM: D

1. {

#include &lt;stdio.h&gt; // q1 states that it is a keyword at margin red

1. }

2. .

```
int | float | char { prints ("Keyword"); } // Q1
[a-zA-Z]* { prints ("Identifier"); } // Q2
, ; { prints ("Separator"); } // Q3
% .
```

```
void main() { // Q4
    if (fscanf (fd, "%d", &num) != P-0) // Q5
        { fprintf (fd, "%d\n", num); } // Q6
    yylex(); // Q7
}
```

Commands to use in terminal:

To create a file → vi filename.l

To save file → esc :wq !(return) then q

lex filename.l

cc lex.yy.c ← Execute file

./a.out ← generate output file.

### PROGRAM : I

09-112009

lex program to identify datatype - int, char, float

→ | OUTPUT:

Enter 04736

Number: 04736

## PROGRAM : 2

S : H 490009

Y. Y. no returns value diff b/w program & code  
. Echo;  
Y. Y.

```
int yywrap (void)
{
    if (yylex() == 2) return 1;
    else return 0;
}
```

## → OUTPUT:

~~Goodmorning~~~~Goodmorning~~

(Output till)

(Inst W)

(or not)

L930

H120

L930 = 280

H120 = -1120

## PROGRAM : 2

Lex program to identify each character as consonant or vowel.

% option noyywrap

%

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

%

%

```
alpha[i] | A | E | I | O | U { printf ("%s=%vowels", yytext); }
```

```
[a-zA-Z] { printf ("%s=%consonants", yytext); }
```

%

```
int main()
```

{

```
    yylex();
```

```
    return 0;
```

}

→ Output

kaushik

au<sup>i</sup> = vowels

kshtk = consonants

## PROGRAM : 3

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

QF

#include

["0-9"]\* {printf ("y.s = digits", yytext);}  
[a-zA-Z]\* {printf ("y.s = characters", yytext);}

→ Output

15 abdc 321

15 321 = digits

abdc = characters

## PROGRAM : 4

lex program to count number of words in an input sentence

Y.Y.

[a-zA-Z0-9]+ {c++ ; }

in { printf (" the count is %d", c); }

Y.Y. { ( lexeme "white-space") token ) \* [P-Z]

{ ( lexeme "newline") token ) \* [S-Z-A-Z-O]

int yywrap() { }

int main()

{

printf (" enter the sentence");

yytex();

return 0;

}

→ Output:

Enter sentence: I'm in BMSCE

Count is : 3

## PROGRAM: 5

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

1.

```
int vowel-count = 0; // for vowel counting
```

```
int const-count = 0; // for consonant counting
```

```
char str[100]; // for string input
```

```
{aeiouAEIOU] {vowel-count++ ;}
```

```
[a-zA-Z] {const-count++ ;}
```

```
ln {printf(" vowel-count=%d , const-count=%d", vowel-count,  
const-count);}
```

```
1. r.
```

→ Output:

Enter: Kaushik

vowel-count=3 , const-count=9

```
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

Q1. 7.

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

7. 7.

→ Output: - abc - abc → "Valid String" → abc → abc

acbd152

invalid string

abc

valid string

123

valid string

## PROGRAM:7

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

%%

```
int{float}char { printf("y.s = keywords", yytext);
[0-9]* { printf("y.s = numbers", yytext);
[a-zA-Z]* { printf("y.s = characters", yytext);
.;
```

void main()

```
{ printf ("Enter f-name"),
scanf("y.s", fname);
yyin = fopen(fname, "r");
yylex();
fclose(yyin);
```

→ Output

Enter f-name: k.txt

Keywords : float, number : 0123, character : kp

## ② PROGRAM:8

Modify program7 such that the output is available in an output file. Use the same sample input.

%%

```
{ fprintf(yyout, "Keywords: y.s", yytext); }
" numbers: y.s"
" characters: y.s"
```

void main()

{

```
printf("Enter input file");
scanf("y.s", fname);
printf("Enter output file");
scanf("y.s", fname);
yyin = fopen(fname, "r");
```

```
yfout=fopen(foname,"w");
```

`yyflex();` function to read input at every step

`fclose(yyin);` closes file with large file size

filere (yyout);

Chlorophyll measured = 23.4 mg/m<sup>3</sup> water depth 11m

→ Output ~~Input~~ ~~Processing~~ ~~Output~~

Enter input file

k.txt

10. *Salvia B. g.*

~~Enter output file (format: .ods) ending~~

~~kout.txt~~ (student, "E:\") n=2

fix: smart (arguably) - very

5 ( ) x-9} \times

(alpha) 2013

*Aug 10*

3rd year; second year rest year

• 2019 : Resümee , Thoth : Innenwelt

8 HAGNAY

1. Standardized test results - Math, English, Science, Social Studies

1998-00-01 12:01, SITE SIGN NO

Digitized by srujanika@gmail.com

2023-08-28 17:49:51 "main" 127.0.0.1

2. *medieval*

• "It's just another day."

Digitized by srujanika@gmail.com

Lorraine Bierwagen

(last update after 1722 hours)

2019-02-28 13:59:23

~~(20002, 184) 102~~

[July 1997] 1997-1998

```
bmscsecse@bmscsecse-OptiPlex-5070:~/Documents/IBM21CS083$ lex p.l  
bmscsecse@bmscsecse-OptiPlex-5070:~/Documents/IBM21CS083$ gcc lex.yy.c  
bmscsecse@bmscsecse-OptiPlex-5070:~/Documents/IBM21CS083$ ./a.out  
enter the input file name  
input.txt  
enter the output file name  
output.txt  
bmscsecse@bmscsecse-OptiPlex-5070:~/Documents/IBM21CS083$ █
```

## PROGRAM:9

LEX Program to recognize floating point numbers.

```

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

```
else { flag = 0; }
```

In if return 0; }

→ Output: How are you today? flag = 0

Enter a Sentence : My Name is Kaustik

~~Simple Sentence~~

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

```
bmscecse@bmscecse-OptiPlex-5070:~/Documents/18M21CS083$ lex float.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/18M21CS083$ gcc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/18M21CS083$ ./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]\* { }

;

ln { return 0; }

%y.

### Output

Enter a sentence : The tower is high

Starting with an article

Enter a sentence : tower is high no tank

Does not start with an article.

### PROGRAM: 12

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

%y.

[a-zA-Z]\* [?!.] fprintf ("ends with punctuation") ;

[a-zA-Z]\* fprintf ("does not end with punctuation") ;

%y.

### Output:

Enter a sentence : The tower is High !

ends with punctuation

Enter a sentence : tower is high

does not end with punctuation

## PROGRAM #13

A. MATAMALA

Let's check if the entered number is signed or unsigned using appropriate meta character.

1.4. Let's check using if a in front of it.

$^1 [+-] [0-9] + \{a=1; \}$

$[0-9] + \{ ? \}$

THM3 28

1.4.

PS THM3 4103A ) "A" "

Output:

Enter i +100 ; +00 ; 0 4103A ) "A" <THM3>

Signed

~~Enter i 100~~

~~Unsigned~~

~~Sept 23~~

( $\epsilon = !$  4000) 16

1.5.  $x = \text{scanf}("x", \&x); \text{if}(x < 0) \text{negative} = \text{true}; \text{else} \text{positive} = \text{true};$

$(\text{"w"}, \&x); \text{if}(x < 0) \text{negative} = \text{true};$

$\text{else} \text{positive} = \text{true};$

1.6.  $b = \text{scanf}("b", \&b); \text{if}(b < 0) \text{negative} = \text{true}; \text{else} \text{positive} = \text{true};$

$\text{if}(a < 0) \text{negative} = \text{true}; \text{else} \text{positive} = \text{true};$

1.7.  $a = \text{scanf}("a", \&a); \text{if}(a < 0) \text{negative} = \text{true}; \text{else} \text{positive} = \text{true};$

### PROGRAM: 14

A 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` & point  
the count in a file called `Output.txt`

```
%* 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 ("In Number of Multiline comments = %.d \n", cc);
    return 0;
}
```

Each token of yyt will be either end of file or

(nonempty, nonwhite)

(W, blank) or get "line"

(with ZZ below it)

(EOF)

(empty)

(blank) giving two

### PROGRAM: 15

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

ip = 1, of 2100 words and return

7.7.

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

7.7.

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:");
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 strings ending in 00.

```

[0-9]*[00] { printf("String is 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 contains 222."); }

```

```

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

```

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(yytext < 5)
```

```
{ printf("y.s doesn't match any rule\n", yytext);
```

```
{ return 2; } // No rule for 5
```

```
else if(yytext[i] == '5')
```

```
{ c++; } // Counting 5's
```

```
for(i=0; i<5; i++)
```

```
{ if(yytext[i] == '5')
```

```
{ c++; } // Counting 5's
```

```
if(c>=2)
```

```
{ printf("y.s matches rule A\n", yytext);
```

```
{ return 1; } // Rule A matched
```

```
}
```

```
if(c<2)
```

```
{ printf("y.s doesn't match any rule\n", yytext);
```

```
break; } // Break from loop
```

```
{ } // End of if(c<2)
```

```
else if(yytext[i-5] == '5') { c--; }
```

```
if(yytext[i] == '5') { c++; }
```

```
{ } // End of if(yytext[i] == '5')
```

```
if(c<2)
```

```
{ printf("y.s doesn't match any rule\n", yytext); }
```

```
break; } // Break from loop
```

```
{ } // End of if(c<2)
```

```
else
```

```
{ } // End of else
```

```
printf("y.s doesn't match any rule\n", yytext);
```

```
{ } // End of printf
```

```
{ } // End of if(yytext[i] == '5')
```

```
yytext[i] = yytext[i+1]; // Shift
```

```
{ } // End of yytext[i] = yytext[i+1];
```

```
return 1; // Rule A matched
```

- 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.

7.7.

$$(1(0)^*(11|01)(01^*01|00^*10(0)^*(11|1))^*\emptyset)$$

$$(11|0(0)^*(11|01)(01^*01|00^*10(0)^*(11|1))^*\emptyset) +$$

print("The set of all strings beginning with a 1  
is interpreted as the binary representation of an  
integer.");

8

7.7.

- e) A set of all strings such that the 10<sup>th</sup> symbol from the right end is 1.

7.7.

$$\{1d\}^*1\{d\}\{9\}^*$$

print("The set of all strings such that the 10<sup>th</sup>  
~~symbol~~ symbol from the right end is 1");

8

7.7.

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

7.7.

```
{d}{d}{d}{d}
int sum=0,i;
for(i=0;i<4;i++)
{
```

    sum+=text[i]-48;

```
}if(sum==9)
{
```

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

for (int i=0; i<4; i++) {  
 for (int j=i+1; j<4; j++) {  
 for (int k=j+1; k<4; k++) {  
 for (int l=k+1; l<4; l++) {

if ((i+j+k+l) == 9) {  
 cout << i << j << k << l << endl;  
 }  
 }  
 }  
 }  
}

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

```
'/i.  
{  
    for (i=0; i<4; i++) {  
        for (j=i+1; j<4; j++) {  
            for (k=j+1; k<4; k++) {  
                for (l=k+1; l<4; l++) {  
                    if (yytext[i] > yytext[i+1]) {  
                        if (yytext[i+2] > yytext[i+3]) {  
                            sum = 0;  
                            break;  
                        }  
                    }  
                }  
            }  
        }  
    }  
}  
if (sum == 1) {  
    printf(" ascending order from left to right.");  
}  
else {  
    printf("y.s doesn't match any rule\n", yytext);  
}  
'/:r.
```

$\Rightarrow$  Output:

a) 10100

string accepted

b) 1222

string accepted

34560 (not in ascending order)

String rejected

String rejected

Sof  
H12/2023

(not in ascending order)

(not in ascending order)

```
bmscecse@bmscecse-OptiPlex-5070: ~/Documents/1BM21CS... Q = - □ ×
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
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ lex re7.l
```

```
fail
bmscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ lex blank.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ ./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/IBM21CS083$ lex re1.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ ./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/IBM21CS083$ lex re2.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ ./a.out
12142
12142 string does not have 222
24322245
24322245 string has 222
```

```
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BH21C5083$ lex re7.l  
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BH21C5083$ gcc lex.yy.c  
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BH21C5083$ ./a.out  
45612  
2fail  
bmscecse@bmscecse-OptiPlex-5070:~/Documents/1BH21C5083$ ./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",
                           "break", "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);  
}  
else if (isalpha(token[0]) || token[0] == '_')  
{  
    printf("Identifier: %s\n", token);  
}  
else if ( strchr("+-*/%<=>()", token[0]) != NULL)  
{  
    printf("Operator: %s\n", token);  
}  
else if ( strchr("();{}.,?;]", token[0]) != NULL)  
{  
    printf("Punctuation: %s\n", token);  
}  
else  
{  
    printf("Invalid: ");  
}  
}  
  
int main()  
{  
    char input[500];  
    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"; analiza();
        analizeToken(token);
        tokenIndex = 0; analiza();
    }
}
if(input[i] != '+' && input[i] != '\n' && input[i] != 't')
{
    if(tokenIndex > 0) analiza();
    token[0] = input[i];
    token[1] = "10"; analiza();
    analizeToken(token);
}
if(tokenIndex == 0) analiza();
return 0;
}

```

=> Output:

$$\text{Sum} = a + b + 10 ?.$$

Identifier: sum

Operator: +

Identifier: a

Operator: +

Identifier: b

Operator: +

Number: 10

Punctuation: ?

Punctuation: ?

10  
10

SB  
18/12/2023

enter c code

```
int a = 1234 ;
```

Keyword: int

Identifier: a

Punctuation/Operator: =

Number: 1234

Punctuation/Operator: ;

## PROGRAM: 18 - Recursive Descent

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

```
char input[100];
```

```
int ind=0;
```

```
void match (char expected)
```

```
{
```

```
    if (input[ind]==expected)
```

```
{
```

```
    ind++;
```

```
}
```

```
3. now output -> ind=0 swizor9 ). Hwing.
```

```
void A();
```

```
void S()
```

```
{
```

```
    match ('c');
```

```
    A();
```

```
    match ('d');
```

```
{
```

```
void AC()
```

```
{
```

```
    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:  
caad\$

Hello

Parsing successful.

(2) Enter the input string:  
caaad\$

Hello

Parsing failed.

(Note: at bottom giving warning  
of extra characters)

```
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);
   |         ^
mscsecse@bmscsecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ^C
mscsecse@bmscsecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ^C
mscsecse@bmscsecse-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);
   |         ^
mscsecse@bmscsecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent
Enter the input string:
aad
Hello
Parsin failed. Extra characters found.
mscsecse@bmscsecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent
Enter the input string:
aad
Hello
Parsin failed. Extra characters found.
mscsecse@bmscsecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent
Enter the input string:
ab$
Hello
Parsin successful.
mscsecse@bmscsecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent
Enter the input string:
aad$
Hello
Parsin failed. Extra characters found.
mscsecse@bmscsecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent
Enter the input string:
abd$
Hello
Parsin successful.
mscsecse@bmscsecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent
Enter the input string:
aad$
Hello
Parsin failed. Extra characters found.
```

## PROGRAM 19: Desk calculator.

```
r. {  
#include "y.tab.h"  
r. }  
r. r.  
[0-9] + { yyval = atoi(yytext); return NUM; }  
[le];  
In return;  
- return yytext[0];  
r.y.
```

```
r. f  
#include <stdio.h>  
r. }  
Y. tokenNUM  
Y. left '+'  
Y. right '-'  
r.y.  
expr: { printf("valid expression\n");  
        printf("Result : %d\n", $1); return 0; }  
e: e + e { $1 = $1 + $3; }  
| e - e { $1 = $1 - $3; }  
| NUM { $1 = $1; }  
;  
Y. y.
```

→ Output:

Enter an arithmetic expression

$5 \cdot 6 + 1$

10

✓ sum  
81124

Leftmost operator

<ArithGeo solving

83

Human friendly

5 \* 6 + 1

31

Leftmost operator (ArithGeo) solving steps

Execution of leftmost operator (ArithGeo)

$$\{ (27+1) \cdot 6 \} + 9 = 3$$

$$28 \cdot 6 + 9 = 167 + 9 = 176$$

$$83 \{ = 176 \} \text{ Human}$$

```
Processing triggers for man-db (2.7.0-1) ...
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$ 
```

## PROGRAM : 20

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

PL-Y

-1- 6

```
#include <math.h> FallBud = auto & first
```

#include <ctype.h> // Standard C header file

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

```
#include <stdlib.h>
```

```
#include <String.h>
```

## Struct tree-node

{ "id": "1", "name": "John Doe", "age": 30, "city": "New York", "isEmployed": true };

char val[10]; // 10 memory

int rc;

3;

`int ind;`

Struct tree-node Syn-tree [100];

void my-print-tree(int cur-ind);

```
int mknodc (int lc, int rc, char val [10]);
```

1.

(post-tax cash or loss) short-term cash

## 7. token digit

7.7. (Inv. Inv. Einheit seit - zw.) - späte

S.E { my - print - tree ( \$1 ) ; } S.A - my2

; ~~sort = gr. This is 7 sort - my 8~~

10

```
E:E '+' T $= mknode ($1,$3,'+') ;;
```

IT  $\{ \$\$ = \text{";} \}$

3

T: T \* E { \$ \$ = mknode( \$ b, \$ 3, "+"); }  
IF { \$ \$ = \$ 1; }  
;

F: ' ( ' E ') ' { \$ \$ = \$ 2; }

1 digit \$ char buf[10]; sprintf( buf, "%d", yyval);  
\$ \$ = mknode( -1, -1, buf );

int main()

{

ind=0;

printf("Enter an expression \n");

yyparse();

return 0;

}

int yyerror()

{

printf("INITIAL Error\n");

if (yychar == '\n') yychar = ' ';

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;

}

```

void my-print-tree (int cur-ind)
{
    if (cur-ind == -1) return; // base case
    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 // current node is left child or right child
        printf(" Operator Node → Index : %d , Value : %s ,
               Left Child Index : %d , Right Child Index : %d
               In " , 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);
}

```

pol

```

% {
    #include "y.tab.h"
    extern int yyval;
    int yylex();
    if (yyval > atoi(yytext); return digit; }

    [ \n ] return 0;
    return yylex[0];
}

```

```

int yywrap()
{
}

```

Output

Enter an expression  $(1 + 2 \times 3) * 4 + 5$

$2 + 4 * 3 + 1 = 2 + [3 * 4] + 1$  sort-mys

(+ mys)

Operator Node  $\rightarrow$  Index: 2, Value: +, left Child  
Index: 0, right Child Index: 1

Digit Node  $\rightarrow$  Index: 0, Value: 2

Digit Node  $\rightarrow$  Index: 1, Value: 4 (2\*)

~~4 \* 3 = 12~~ sort-mys

~~12 + 1 = 13~~ sort-mys

~~(13 + 5) sort-mys~~

$(2 * [3 * 4] + 1) * 5$  sort-mys

$(2 * [3 * 4] + 1) * 5$  sort-mys

19

33

"stack" sort-mys

Slow type sort-mys

84

"right operator" sort-mys

10

"a operator" sort-mys

(D07) sort-mys

19 sort-mys

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\$

## PROGRAM: 2

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

String.1 (a<sup>n</sup>) string) n= 992 AAAAAA : line 2

Y. { d(d<sup>n</sup>) star .

#include <Statio.h> /& n=2 = 5 or

#include <Stdlib.h>

#include "y.tab.h"

extern int yyval;

Y. ?

Y. Y.

[aA] { yyval = yytext[0]; return A; }

[bB] { yyval = yytext[0]; return B; }

In { return NL; }

{ return yytext[0],? } star ) string

Y. Y.

int yywrap()

{

return 1;

}

((n1 + n2) \* n3) nnnyy - t1

String. Y

Y. {

#include <Statio.h>

#include <Stdlib.h>

int yyerror (char \*s);

int yylex (void);

Y. ?

Y. token A (d<sup>n</sup>) star n= 992 99999

Y. token B

Y. token NL

double t1

100.000

100.000

ddDD

99.99999

18 NOV-2019

1.7. 750 serial of 5 tokens given  
Smtx: AAAAASBB NL { printf (" Passed using the  
rule  $(a^n)b$ ,  
 $n \geq 5$ . In Valid String \n ); }

s: SA

1

;

;

1.7. A pointer to function = tokenif } PA 07  
void main() { P07 tokenif = tokenif } P07  
{  
printf (" Enter a String \n "); /\* input \*/  
yyerror();  
}

int yyerror (char \*S)  
{

printf (" Invalid String ! \n ");  
return 0;

}

=> Output

Enter a String!  
aaaaab

Parsed using the rule  $(a^n)b, n \geq 5$ .  
Valid String

aabb

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
Enter a string!
abb$
invalid String!
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
Enter a string!
abb
invalid String!
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
Enter a string!
aaab
invalid String!
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ ./a.out
Enter a string!
aaaab
parsed using the rule (an)b, n>=5.
valid String!
aaaaaab
invalid String!
mscecse@bmscecse-OptiPlex-5070:~/Documents/1BM21CS083$ 
```

## PROGRAM : 22

## Infix to Postfix

-2-

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

```
#include <stdlib.h>
```

#include "y.tab.h"

extern int yylval;

۱۴

8. (cont'd) Using } 9:21

16

[0-9]+ { yyval = atoi(yytext); return num; }  
[tE]

1

```
In y return 0; }  
{ return yytext[b]; }
```

```
{ return yytext[b]; }
```

۷۷

```
int yywrap () { if (fopen ("b.o", "w")) return 1; }
```

## Prefix to Postfix .y

• C. }

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

#include <stdlib.h>

int yerror (const char \*s);

~~int yylex (void);~~

4.3

~~(nicht-lösung) nicht lösbar~~ Lösbar

4. token num

4. left  $\leftarrow \leftarrow$

4. test '\*' '/'

4. left - )

4. left 'c'

7.

s: e { printf ("\n"); }

;

e: e '+' t { printf ("+"); } (additive structure)

| e '-' t { printf ("-"); } (additive structure)

t

;

t: t '\*' h { printf ("\*"); }

| t '/' h { printf ("/"); }

\n

error message (fscanf): old = input(p) + P - qT

E-17

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

if

  | p: P-1 fscanf (marker)

;

f: 'c' ~'

lnum { printf ("%d", \$1); } () grammar fail

;

.7.

void main()

{

printf ("Enter an infix expression: \n");

yyparse();

}

int yyerror (const char \*s) (additive structure)

{ (2.3 marks times) reasoning fail

printf ("invalid infix expression\n"); fail

return 0

?



```
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.1

Y.9

d [0-9]+

a [a-zA-Z]+

Y.Y.

{d} {yyval = atoi (yytext); return digit; }

{a} {strcpy (iden, yytext); yyval = 1; return id; }

[1T] {~~if string mit id startet~~ A72 \* 888}

In return 0;

\* return yytext[0];

Y.Y.

Addressed.v

Y.Y.

S: id' = 'E' {printf ("t.s=t%.d\n", den, var-cnt);}  
E: E '+' T { \$\$ = var-cnt; var-cnt++; printf ("t%d  
|E| - T = t%." );

:

T: T '\*' F { " — " — " — " → " " } ;

L T Y F { " — " — " — " → " " }

IF { \$\$ = \$1; }

;

Y.Y.

Output



Enter an expression

$$a = 8 + 9 - 2$$

$$t_0 = 8;$$

$$t_1 = 9;$$

$$t_2 = t_0 + t_1;$$

$$t_3 = 2;$$

$$t_4 = t_2 - t_3;$$

$$a = t_4$$

29/11/2024

```
mscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ lex 3addcode.l
mscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ yacc -d 3addcode.y
mscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ gcc lex.yy.c y.tab.c
mscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ ./a.out
```

Enter an expression:

```
=8+9-2
0 = 8;
1 = 9;
2 = t0 + t1;
3 = 2;
4 = t2 - t3;
=t4
```

```
mscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ ./a.out
```

Enter an expression:

```
=2^3/23+5
0 = 2;
1 = 3;
2 = t0 ^ t1;
3 = 23;
4 = t2 / t3;
5 = 5;
6 = t4 + t5;
=t6
```