

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT

on

COMPILER DESIGN

Submitted by

JAGADEESH A LATTI(1BM21CS079)

Under the Guidance of
Prof. Prameetha Pai
Assistant Professor, BMSCE

in partial fulfilment 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

November 2023-February 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 **JAGADEESH A LATTI(1BM21CS079)**, who is bonafide student of **B. M. S. College of Engineering**. It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2023-24.

The Lab report has been approved as it satisfies the academic requirements in respect of **Compiler Design- (22CS5PCCPD)** work prescribed for the said degree.

Prof. Prameetha Pai
Assistant professor
Department of CSE
BMSCE, Bengaluru

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

B. M. S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND

ENGINEERING



DECLARATION

I, Jagadeesh A Latti(1BM21CS079), student of 5th Semester, B.E, Department of Computer Science and Engineering, B. M. S. College of Engineering, Bangalore, here by declare that, this lab report entitled "**Compiler Design**" has been carried out by me under the guidance of Prof. Sunayana S, Assistant Professor, Department of CSE, B. M. S. College of Engineering, Bangalore during the academic semester November-2023-February-2024.

I also declare that to the best of my knowledge and belief, the development reported here is not from part of any other report by any other students.

TABLE OF CONTENTS

INDEX

Name : Jagadeesh A.L. Class : 5-B
Section : Roll No. : Subject : CD-LAB

Sl. No.	Date	Title	Page No.	Teacher Sign. / Remarks
		Lex	9/10	(B)
1		Write a lex program to count no of words in input		
2		Write a lex program to count no. of vowels & consonants		
3		Write a lex program to identify alphabets as character inventory		
4		Write a program in LEX to recognise float point numbers		
5		Write a lex program that copies a file, replacing each non-empty space by white space		
6		The set of all strings end with 00		
7		Write a program to design lexical Analysis		

PROGRAM-1

- 4) Write a Lex program to count no. of words in input sentence.

/*-*/

[a-zA-Z]* {count++;}

in & point ("No of words: %d\n", count);

/*-*/

Output:

Jagga is here.

No. of words: 3

- 5) Write a Lex program to count no. of vowels & consonants in a string.

/*-*/

aeiolovelAEIOLUV {countv++;}

[a-zA-Z] {countc++;}

in & point ("No of vowels: %d\n", countv);

point ("No of consonants: %d\n", countc);

/*-*/

Output:

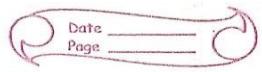
Good morning

No of vowels: 4 No of consonants: 7

OUTPUT SCREENSHOT:

```
Enter a sentence:  
Compiler design  
    No of vowels and consonants are 5 and 9  
This is a book  
    No of vowels and consonants are 5 and 6  
AC
```

PROGRAM-2



3) Write a Lex program to identify alphabets or characters & numbers as digits.

-i. option nyywoap
-j. d

#include <stdio.h>

-k. y

-l. z

[0-9]* l point ("y.y: digit\n", yytext); }
[a-zA-Z]* l point ("y.y: alphabets\n", yytext); }
-m. y.

void main()

{

lex();
}

Output :

good evening:17-10-2017 is a date

17: ~~alphabets~~ digits, 17, 10, 2017

good evening: alphabets 17, 10, 2017

17: ~~alphabets~~ digits, 17, 10, 2017

4) Write a Lex program to count no. of words
in a input sentence.

/*-*/

[a-zA-Z]* {count++;}

in & point ("No of words", &ln, (count));

/*-*/

Output:

Jaggu is here,

No. of words: 3

5) Write a Lex program to count no. of vowels &
consonants in a string.

/*-*/

aieeiolahelAEIIOUV {countv++;}

[a-zA-Z] {countc++;}

in & point ("No of vowels", &ln, countv);

point ("No of consonants", &ln, countc);

/*-*/

Output:

Good morning

No of vowels: 4 No of consonants: 7

OUTPUT SCREENSHOT:

```
Enter a sentence:  
Compiler design  
    No of vowels and consonants are 5 and 9  
This is a book  
    No of vowels and consonants are 5 and 6  
AC
```

```
Enter a sentence:  
My name is Neha  
    No of words in the sentence are 4.  
I will make things happen.  
    No of words in the sentence are 5.
```

PROGRAM-3:

Date _____
Page _____

- (Q) Write a program in LEX to recognize Floating Point Number. Check for all the following input cases.

-1.2

#include <stdio.h>

1.2

-1.2

1[-+]?[0-9]*[.]?[0-9]+ of point ("1.2 is floating point number in", yytext);
1[-+]?[0-9]* of point ("1.2 is not floating point number", yytext);

-1.2

int yywrap()

{

y

void main()

{

point("Enter a number\n");

yylex();

y

Output: 0.33

0.33 is floating point number.

-123

-123 is not floating point number.

+9.6

+9.6 is floating point number

OUTPUT SCREENSHOT:

```
Enter a number:  
23  
Not a floating point number!  
  
0.5  
Floating point number!  
  
.8  
Floating point number!  
  
-.9  
Floating point number!  
  
+56  
Not a floating point number!
```

PROGRAM-4

Date _____
Page _____

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

1. d

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

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

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

```
char string[100];
```

2. y

3. -

```
[in] &tprintf(yyout, "%s\n", string); string[0] = '0';
```

```
[ ] &tprintf(yyout, "%s", string), string = '0';
```

```
&tprintf(yyout, "%s", ""); }
```

```
strcat(string, yytext);
```

```
<<FOR>>&tprintf(yyout, "%s", string), return 0; }
```

4. "

int main()

{

```
char filename[100];
```

```
printf("Enter the name of the file to copy: \n");
```

```
scanf("%s", filename);
```

```
yyin = fopen(filename, "r");
```

```
if (yyin == NULL)
```

2

```
printf("Can't open file %s\n", filename);
```

```
exit(1);
```

3

```
yylex();
```

8/12/23

ion yywrap(void)

{

 return 1;

}

Output:

Enter the name of the file to copy: in

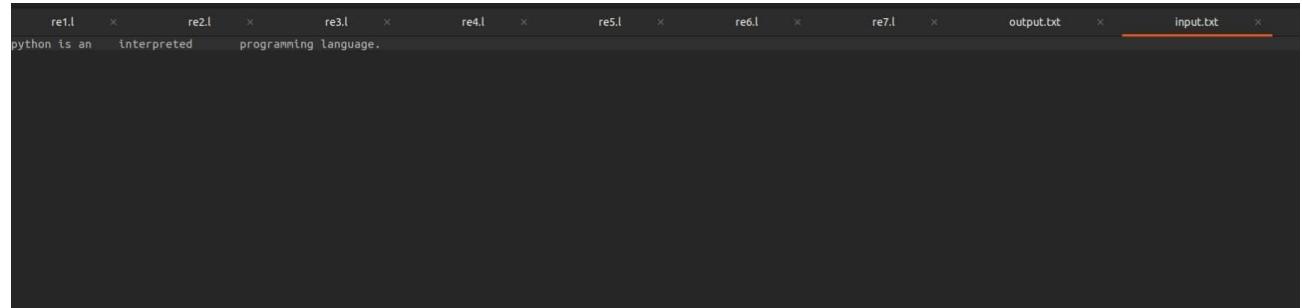
Enter the name of the file to open for
writing: out

[0-9][0-9][0-9][0-9]

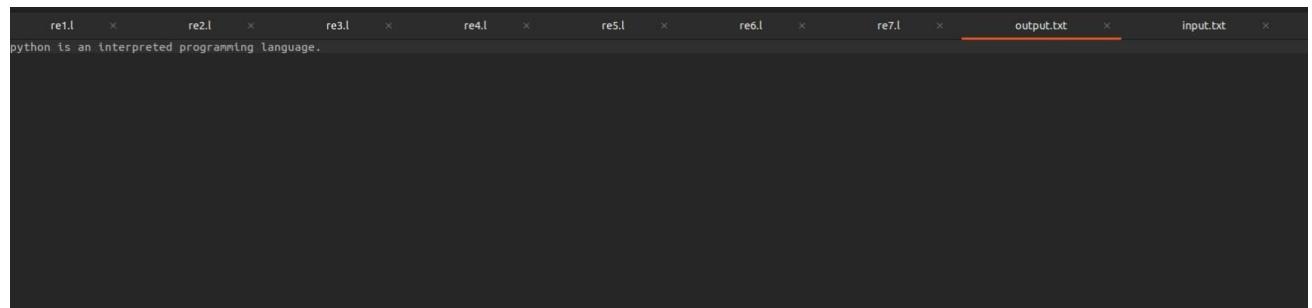
[0-9A-Z]* L point ("invalid input");

OUTPUT SCREENSHOT:

```
Enter the name of the file to copy:      input.txt
Enter the name of the file to write:     output.txt
```

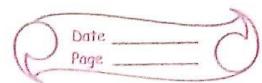


A screenshot of a terminal window showing the contents of a file named 'input.txt'. The file contains the text: "python is an interpreted programming language." The terminal has several tabs open at the top, including 're1.l', 're2.l', 're3.l', 're4.l', 're5.l', 're6.l', 're7.l', 'output.txt', and 'input.txt' (which is currently active).



A screenshot of a terminal window showing the contents of a file named 'output.txt'. The file contains the text: "python is an interpreted programming language." The terminal has several tabs open at the top, including 're1.l', 're2.l', 're3.l', 're4.l', 're5.l', 're6.l', 're7.l', 'output.txt' (which is currently active), and 'input.txt'.

PROGRAM-5



a) the set of all strings ending in 00

-1. option anywarrp

-1. x

#include <stdio.h>

1. y

1. -1.

[0-9]*00 & point ("String accepted"); }

[0-9]* & point ("String not accepted"); }

1. -1.

void main()

2

point ("Enter string : ");

getchar();

x

output:

001

String rejected

100

strong accepted

b) the set of all strings with three consecutive 'm's.

option noyywrap

i. L

ii. include <stdio.h>

j. y

k. j.

[0-9]* 222[0-9]* l point ("String accepted")

[0-9]* l point ("String not Accepted")

j. j.

void main()

{

point ("Enter String: ");

yytext;

}

Output:

1234

String rejected

Output

String accepted

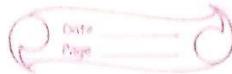
SF
11/12/2023

OUTPUT SCREENSHOT:

```
bmscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ ./a.out
1111
successbmscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ ./a.out
11
bmscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ lex re5.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ 
bmscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ ./a.out
1023002245
1023002245 10th symbol from right end id 1
^Z
[1]+ Stopped ./.a.out
bmscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ lex re6.l
bmscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ cc lex.yy.c
bmscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ ./a.out
9000
success
bmscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ ./a.out
4005
success
bmscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ ./a.out
123
123fail
bmscecse@bmscecse-OptiPlex-5070:~/Documents/IBM21CS083$ lex re7.l
```

```
1111
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
[]
```

PROGRAM-6



Write a program to design lexical Analyzer in C.

Q.

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

int iskeyword(char *s1)
{
    char keywords[8][10] = {"int", "float", "if",
                           "else", "while"};
    int i;
    for (i=0; i<5; ++i)
        if (strcmp(keywords[i], s1) == 0)
            return 1;
    return 0;
}
```

```
int isOperatorOrPunctuation(char ch)
```

```
{
    char operators[] = "+-*%/=";
    char punctuation[] = "(),{}[]";
    int i;
    for (i=0; i<8; ++i)
        if (operators[i] == ch)
            return 1;
    return 0;
}
```

```
for (i = 0; isSpace(punctuation[i]); ++i)
```

```
    if (punctuation[i] == 'h')  
        return 1;
```

```
    return 0;
```

```
void lexicalAnalyzer(char *input)
```

```
    int i = 0;
```

```
    int len = strlen(input);  
    while (i < len)
```

```
        if (!isPunc(input[i]))
```

```
            i++;
```

```
        continue;
```

```
        if (isAlpha(input[i]) || input[i] == '-' ||
```

```
            input[i] == '_')
```

```
            char token[50];
```

```
            int j = 0;
```

```
            token[j] = input[i++];
```

```
            while (isAlnum(input[i]) ||  
                input[i] == '-' ||
```

```
                input[i] == '_')
```

```
            token[j] = input[i++];
```

```
        }
```

```
        token[j] = '\0';
```

Date _____
Page _____

```
if (isKeyWord(token))  
    point("key word : %s, token");  
else  
    point("Identifier : %s, token");  
continue;  
if (isDigit(input[i]))  
    char token[50];  
    int j=0;  
    token[i] = input[i];  
    while (isDigit(input[i]))  
        token[i+j] = input[i+j];  
    i++;  
int main()  
{  
    char input[1000];  
    point("Enter the input strings");  
    gets(input, sizeof(input), stdin);  
    point('Tokenizing the input : \n');  
    lexicalAnalyzer(input);  
    return 0;  
}
```

Output:

Enter the input string: int a=0;

Tokenizing the input:

Key word: int

Identifier: a

operator or Punctuation: =

Number: 0

operator or punctuation: ;

✓ 8/2/23
18/2/23

OUTPUT SCREENSHOT:

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

PROGRAM-7

Date _____
Page _____

Write a Program to perform Recursive Descent parsing on the following grammar.

$$S \rightarrow aAa, A \rightarrow abAb$$

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

```
bool parse_S(char input -&input[ ]);  
bool parse_A(char input -&input[ ]);  
bool recursive_descent_parser(char input -&input[ ]);  
int index;  
*
```

```
bool parse_S(char input -&input[ ]) {
```

```
}
```

```
if (input -&input[index] == 'c') {
```

```
    *
```

~~Non-terminal~~

```
: & (parse_A(input -&input) && input -&input == 'd')
```

```
    *
```

```
    index++;
```

```
    return true;
```

```
}
```

```
else
```

```
    return false;
```

```
}
```

bool parse_A (char input [])

if (input [index] == 'c')

if (input [index] == 'b')

*

index + 1;

*

return true;

*

else

return false;

bool recursive_descent (char input [])

*

if (parse_A && input == '\0')

*

return true;

*

else

return false;

int main()

*

char user_input [100];

printf ("Enter a string to parse: ");

scanf ("%s", user_input);

printf ("A -> (Ad \n");

printf ("A -> (b\ln \n");

it (recursive - discuss - input)

2

point ("The given string is accepted")

y

else

L

point ("The given string is not accepted")

u

so

Output:

Enter a string to parse: cd

S → CA

A → abla

The given string is not accepted by grammar.

Enter a string to parse: cabd

S → CA

A → abla

The given string is accepted by grammar.

OUTPUT SCREENSHOT:

```
nter the input string:  
ad  
ello  
arsing failed. Extra characters found.  
mscse@bmscsecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent  
nter the input string:  
aad  
ello  
arsing failed. Extra characters found.  
mscse@bmscsecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent  
nter the input string:  
ab$  
ello  
arsing successful.  
mscse@bmscsecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent  
nter the input string:  
aad$  
ello  
arsing failed. Extra characters found.  
mscse@bmscsecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent  
nter the input string:  
abd$  
ello  
arsing successful.  
mscse@bmscsecse-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ./recursive_descent  
nter the input string:  
aaad$  
ello  
arsing failed. Extra characters found.
```

Design a suitable grammar for evaluation of arithmetic exps. having + & - operators. + has left & left associativity.

Code:

```
#option noyywrap  
%{  
#include "y.tab.h"  
%}  
[0-9]+ lval = atoi(yytext); return NUM;  
[+ -];  
{ or return V;  
    return yytext[0];  
}- -;
```

%{

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

```
%}{
```

```
y.ytok = NUM;
```

```
y.left = '+';
```

```
y.right = '-';
```

```
y.j =
```

```
exp();
print("Valid expression\n");
```

```
printf("Result: %d\n", $); return 0;}
```

```
e: e '+' e & $ = $1 + $3; }
```

```
1 e '-' e & $ = $1 - $3; }
```

```
1 Num & $ = $1; }
```

```
;
```

```
int main()
```

```
{
```

```
    point("Enter an arithmetic expression");
```

```
    yyparse();
```

```
    return 0;
```

```
}
```

```
int yyparse()
```

```
{
```

```
    point("Invalid expression");
```

```
    return 0;
```

```
}
```

Output:

Enter an expression

5 + 10 - 2

Valid expression

Result: 13.

~~Don't
forget~~

OUTPUT SCREENSHOT:

```
Enter an arithmetic expression:  
2+3*4  
Valid expression!  
Result:14
```

```
Enter an arithmetic expression:  
2++3-  
Invalid expression!
```

PROGRAM-9

Write your program stores program "Forward".
 $a^n b^n \rightarrow S$.

if. ok

#include <cs1d.h>

#include <cs1lib.h>

int yyerror (char *s);

int yylex (void);

if.

if. token A

if. token B

if. token NL

if-1 if

Errno: AAABAFBNAEL points ("parsed using the
rule ($a^n b^n \rightarrow S$) is valid string")

;

8, SA

1

;

if-1.

void main()

if

point ("Enter a string 1, 1a");

yyparse();

,

int yyerror (char *s)

2

point (" Invalid string in");

)

y. d

#include < stdio.h >

#include < stdlib.h >

extern int yyval;

-J-)

-L X

[ca] & yyval = yytext[0]; return A;)

[cb] & yyval = yytext[0]; return B;)

10 & yyval = yytext[0]; return NL;)

& yyval = yytext[0];)

-J-).

int yywrap()

L

return 1;

Y

Output:

Enter a string:

aaaaab

pushed using the rule (a^n)b, n>=5

valid string:

aaaaab

invalid string:

8
29/2024

OUTPUT SCREENSHOT:

```
Enter a string!
aaaaaaab
Parsed using the rule (a^n)b, n>=5.
Valid String!
ab
Invalid String!
```

```
Enter a string!
abc
Invalid String!
```

PROGRAM-10

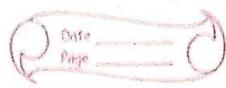
Write a Yacc program to generate syntax tree for a given arithmetic expression.

p1.l

```
'%d
#include "y.tab.h"
extern int yyval;
%}
%t
[0-9]+ yyval = atoi(yytext); return Digit;
%}
%n return 0;
return yytext[0];
%}
int yywrap()
{
%
```

p1.y

```
'%d
#include <math.h>
#include <ctype.h>
#include <stdio.h>
struct tree_node
{
    char val[10];
    int lc;
    int rc;
};
```



int ind;

struct tree-node syn-tree[100];

Void mknodc (int lc, int rc, char val[10])
{ . }

7. token Digit

7. 7.

8. E of my-point-toe (\$1);

E : E' + T & \$8 = mknodc (\$1, \$3, "+"); ; ; ;

1 T & \$8 = S1 ; ; ;

; ; ;

-1. -1.

int main()

{

ind=0; // to indicate the start of expression

printf ("Enter an expression(D):

yyparse();

return 0;

}

int yyerror();

{

printf ("NYTIN Error \n");

y

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

2

strcpy (&syn-tree[ind].val, val);

syn-tree[ind].lc = lc;

ind++;

y

```

void my-point-tree (int cur-ind)
{
    if (cur-ind == -1) return;
    if (syn-tree[cur-ind].l == -1 & syn-tree[cur-ind].r == -1)
        printf ("Digit + Node -> Index : %d, Value : %d\n",
                cur-ind, syn-tree[cur-ind].val);
    else
        pointst ("Operator Node -> Index : %d, Value : %g",
                 my-point-tree (syn-tree[cur-ind].l));
        my-point-tree (syn-tree[cur-ind].r);
}

```

Output:

Enter an expression,

$2 + 3 * 5$

Operator Node -> Index : 4, Value : +, left
 child : 0, Right child : 3

Leaf Node -> index : 0, value : 2

operator Node -> index : 3, value : *, left child : 1,
 right : 2

Leaf Node -> index : 1, value : 3

Leaf Node -> index : 2, value : 5

OUTPUT SCREENSHOT:

```
Enter an expression:  
2*3+5*4  
Operator Node -> Index : 6, Value : +, Left Child Index : 2,Right Child Index : 5  
Operator Node -> Index : 2, Value : *, Left Child Index : 0,Right Child Index : 1  
Digit Node -> Index : 0, Value : 2  
Digit Node -> Index : 1, Value : 3  
Operator Node -> Index : 5, Value : *, Left Child Index : 3,Right Child Index : 4  
Digit Node -> Index : 3, Value : 5  
Digit Node -> Index : 4, Value : 4
```

PROGRAM-11

Date _____
Page _____

Use YACC to convert infix to Postfix express.

part 1

y.yf

#include "y.tab.h"

extern int yyval;

-/->

-/->.

[0-9]+ & yyval = atoi(yytext); return digit; }

[+];

[in] return 0;

return yytext[0];

-/->.

int yywrap()

2

y

part 2

y2

#include <ctype.h>

#include <stdio.h>

#include <stroib.h>

-/->

-/-> token digit

-/->.

E : E & point ("in"); }

:

F : E & '+' & point ("+"); }

```
    T; T ::= digit | char ('n')
```

```
    F: '(' E ')'
```

```
    L digit & print ('1.0', $1);
```

```
;
```

```
digit
```

```
int main()
```

```
{
```

```
    printf ("Enter infix expression: ");
```

```
    yyparse();
```

```
}
```

```
yyerror()
```

```
{
```

```
    printf ("Error");
```

```
}
```

Output:

Enter infix expression: $2+6 \times 3+7$

$2\ 6\ 3\ * + +$

OUTPUT SCREENSHOT:

```
Enter an infix expression:  
2+3*8/4^3-3  
238*43^/+3-
```

PROGRAM-12

```
p1, y  
i, l  
if include <math.h>  
if include <csyph>  
int var (n=0);  
char ident(20);  
y, y  
y, token_id  
y, token digit  
y, -y.  
S: id '=' EXP pointf("y-y = +r-d\n", id, var, cnt);  
E: E '+' T & $& = var -cnt; var -cnt++; $&, $1, $2  
y  
I F '-' T & $& = var -cnt; var -cnt++; $&, $1, $2  
;  
T: T '*' F & $& = var -cnt; var -cnt++; $&, $1, $2  
I T '/' F & $& = var -cnt; var -cnt++; $&, $1, $2  
F: P '^' F & $& = var -cnt; var -cnt++; $&, $1, $2  
I P & $& = $1, y  
;  
'  
P: '(' E '^' 2 & $& = 82; y  
1 digit & $& = var -cnt; var -cnt++; pointf("y-y:  
    rd; ln" $&, $1); y  
;  
y, -y.  
int main()  
{  
    var cnt=0;  
    pointf("Enter an expo n:\n");  
    syscanf();  
    return();  
y}
```

yy errors()

point ("origin")

Output:

Enter an expression:

$$a = 2 + 3^* 6$$

$$t_0 = 2$$

$$x_1 = 3;$$

$$\underline{+ 2 = 6 ;}$$

$$t_3 = t_1 + t_{2i}$$

$$t_2 = t_0 + t_3;$$

$$a = \cancel{5x}$$

$\text{g} \rightarrow \text{g} + \text{g}$

OUTPUT SCREENSHOT:

```
henry29@henry-VirtualBox: /Documents/EXACLB01/Logs/ans> ./a.out
Enter an expression:
a=2*3/6-4
t0 = 2;
t1 = 3;
t2 = t0 * t1;
t3 = 6;
t4 = t2 / t3;
t5 = 4;
t6 = t4 - t5;
a=t6
```

