

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT

on

COMPILER DESIGN

Submitted by

NEHA BHASKAR KAMATH(1BM21CS113)

Under the Guidance of
Prof. Sunayana S.
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 **Kanjika Singh (1BM21CS086)**, 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- (22CSSPCCPD)** work prescribed for the said degree.

Prof. Sunayana S.
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, Kanjika Singh (1BM21CS086), 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

Lab No	Title	Page No
1		6-8
1.1	Write a program in LEX to recognize different tokens: Keywords, Identifiers, Constants, Operators and Punctuation symbols.	6
1.2	Write a program in LEX to count the number of characters and digits in a string.	7
	Write a program in LEX to count the number of vowels and consonants in a string.	8
2		9-15
2.1	Write a program in lex to count the number of words in a sentence.	9
2.2	Write a program in lex to demonstrate regular definition.	10
2.3	Write a program in lex to identify tokens in a program by taking input from a file and printing the output on the terminal.	11-12
2.4	Write a program in lex to identify tokens in a program by taking input from a file and printing the output in another file.	13-14
2.5	Write a program in lex to find the length of the input string.	15
3		16-23
3.1	Write a program in LEX to recognize Floating Point Numbers.	16
3.2	Read and 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.	17
3.3	Write a program to check if the input sentence ends with any of the following punctuation marks (?, fullstop , !)	18-19
3.4	Write a program to read an input sentence and to check if the sentence begins with English articles (A, a,AN,An,THE and The).	20-21
3.5	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 and print the count in a file called output.txt.	22
3.6	Write a program to read and check if the user entered number is signed or unsigned using appropriate meta character.	23
4		24-36
4.1	Write a LEX program that copies a file, replacing each nonempty sequence of white spaces by a single blank.	24-25
4.2	Write a LEX program to recognize the following tokens over the alphabets {0,1,..,9}	26-36

4.2.1	The set of all string ending in 00.	26
4.2.2	The set of all strings with three consecutive 222's.	27
4.2.3	The set of all string such that every block of five consecutive symbols contains at least two 5's.	28-29
4.2.4	The set of all strings beginning with a 1 which, interpreted as the binary representation of an integer, is congruent to zero modulo 5.	30-31
4.2.5	The set of all strings such that the 10th symbol from the right end is 1.	32
4.2.6	The set of all four digits numbers whose sum is 9.	33-34
4.2.7	The set of all four digital numbers, whose individual digits are in ascending order from left to right.	35-36
5		37-38
5.1	Write a C program to design lexical analysis to recognize any five keywords, identifiers, numbers, operators and punctuations.	37-38
6		39-40
6.1	Write a program to perform recursive descent parsing on the following grammar: S->cAd A->ab a	39-40
7		41-47
7.1	Write a program in YACC to design a suitable grammar for evaluation of arithmetic expression having +, -, *, and /.	41-42
7.2	Write a program in YACC to recognize strings of the form $\{(a^n)b, n \geq 5\}$.	43-44
7.3	Write a program in YACC to generate a syntax tree for a given arithmetic expression.	45-47
8		48-49
8.1	Write a program in YACC to convert infix to postfix expression.	48-49
9		50-52
9.1	Write a program in YACC to generate three address code for a given expression.	50-52

Lab 1

1.1 Write a program in LEX to recognize different tokens: Keywords, Identifiers, Constants, Operators and Punctuation symbols.

Code:

Date : 16/11/23

Program 1

To identify Keywords, identifiers and separators in a C program

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

void main()
{
    yylex();
}
```

Program 2

Lex program to identify whether the entered input is a number, operator or invalid character.
It should ignore white space, tab space

Output

```
kanjika@ubuntu:~/Desktop/lab practice$ lex lab1a.l
kanjika@ubuntu:~/Desktop/lab practice$ gcc lex.yy.c
kanjika@ubuntu:~/Desktop/lab practice$ ./a.out
Enter the text int main a 9;
int - Identifier
main - Keyword
a - Identifier
9 - Digit
;- Separator
```

1.2 Write a program in LEX to count the number of characters and digits in a string.

Code

The image shows handwritten code on lined paper. At the top left, there is a red checkmark next to the word "hello". Below it, the text "Program to count digits and characters" is written. The code itself is as follows:

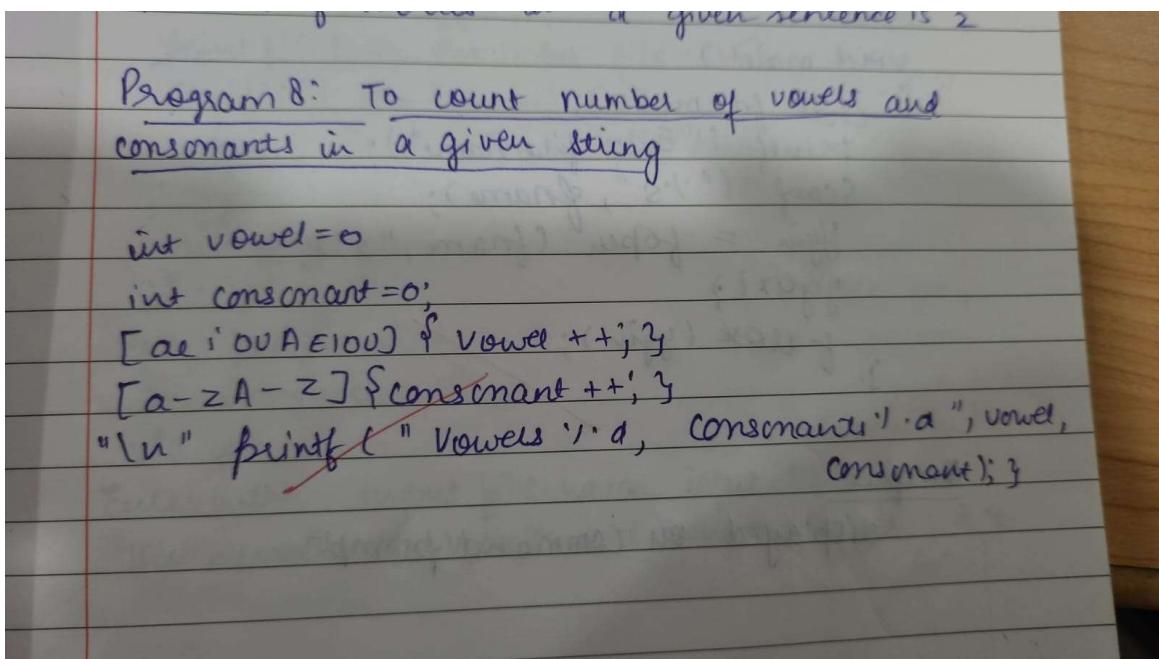
```
int count c=0, d=0; /*  
*/  
[A-zA-Z]+ { word c++; }  
[0-9]+ { d++; }  
In { printf ("Digits - %d, Character %c", d, c), c=0  
};
```

Output

```
kanjika@ubuntu:~/Desktop/lab practice$ lex week1_countDigitsandCharacters.l  
kanjika@ubuntu:~/Desktop/lab practice$ gcc lex.yy.c  
kanjika@ubuntu:~/Desktop/lab practice$ ./a.out  
Enter input  
hey, my address is 6 main road, 221005  
The number of digits in the sentence is 7 and characters is 22
```

1.3 Write a program in LEX to count the number of vowels and consonants in a string.

Code



Output

```
kanjika@ubuntu:~/Desktop/lab practice$ lex week1_NoofVowelsConsonants.l
kanjika@ubuntu:~/Desktop/lab practice$ gcc lex.yy.c
kanjika@ubuntu:~/Desktop/lab practice$ ./a.out
Enter input
hello my name is kanjika
The number of vowels in the sentence is 8 and consonants is 12
```

Lab 2

2.1 Write a program in lex to count the number of words in a sentence.

Code

Date : _____

Program7: To count number of words in input sentence

```
int words=0;
yytext();
[a-zA-Z0-9]* $w++; }
"\n".$ printf (" Number of words in given sentence
is %d",w); }

void main()
{ yytext();
}

Output
hello there
Number of words in a given sentence is 2
```

Output

```
kanjika@ubuntu:~/Desktop/lab practice$ lex week2_noOfwords.l
kanjika@ubuntu:~/Desktop/lab practice$ gcc lex.yy.c
kanjika@ubuntu:~/Desktop/lab practice$ ./a.out
Enter input
they this is test input
The number of words in sentence is 5
this is life
The number of words in sentence is 3
```

2.2 Write a program in lex to demonstrate regular definition.

Code

Programs
numbers as
To identify digits and alphabets as characters

```
[+ -]?[0-9]* { printf("Digits"); }  
[a-zA-Z]* { printf("Alphabet-character"); }
```

Output
enter
Hello123
Digits - 123
Alphabet - character - Hello

Output

```
kanjika@ubuntu:~/Desktop/lab practice$ lex week2_identifyCharDigits.l  
kanjika@ubuntu:~/Desktop/lab practice$ gcc lex.yy.c  
kanjika@ubuntu:~/Desktop/lab practice$ ./a.out  
Enter input  
hello  
characters  
  
78  
Digits  
  
gwhdh982  
Invalid input
```

2.3 Write a program in lex to identify tokens in a program by taking input from a file and printing the output on the terminal.

Code

Read input from file and print on terminals

```
char  
void main()  
{ char fname[10];  
    printf ("Enter file name");  
    scanf ("%s", fname);  
    yyin = fopen (fname, "r");  
    yylex();  
}
```

Output

displayed on command prompt

Output

This file "/home/kanjika/Desktop/lab practice/input.txt" is already open in another window.
Do you want to edit it anyway?

```
1 int x=2 , int y=3;
2 int product=x*y;
```

```
kanjika@ubuntu:~/Desktop/lab practice$ lex week2_Fileinput.l
kanjika@ubuntu:~/Desktop/lab practice$ gcc lex.yy.c
kanjika@ubuntu:~/Desktop/lab practice$ ./a.out
int is a keyword.
x is an identifier.
= is an assignment operator.
2 is/are digit(s).
, is a separator.
int is a keyword.
y is an identifier.
= is an assignment operator.
3 is/are digit(s).
; is a delimiter.
int is a keyword.
product is an identifier.
= is an assignment operator.
x is an identifier.
* is a binary operator.
y is an identifier.
; is a delimiter.
```

2.4 Write a program in lex to identify tokens in a program by taking input from a file and printing the output in another file.

Code

Program 11

Read input from file but output stored in another file. Ask for output file name

```
yyin = fopen (fname, "r");
yyout = fopen (ofname, "w");
yylex();
fclose (yyin);
fclose (yyout);
return 0;
```

Program 12

```
int main () {
    char fname[20];
    extern FILE *yyin;
    printf ("Enter the input file : \n");
    scanf ("%s", &fname);
    yyin = fopen ("fname", "r");
    yylex();
    return 0;
}
```

lex programs.l.

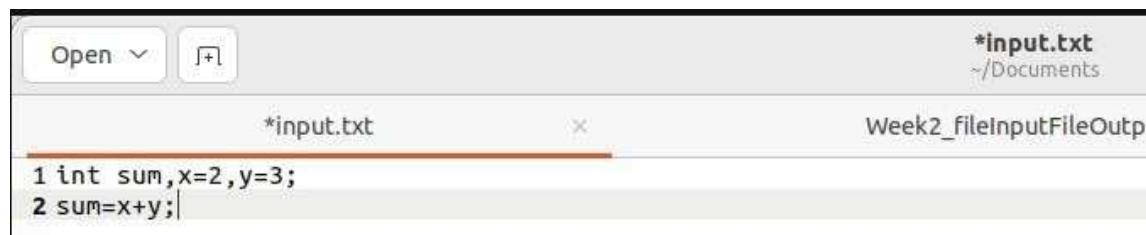
gcc lex.yy.c

./a.out

Enter the input file name input.txt

The number of vowels and consonant is 1 & 9

Output

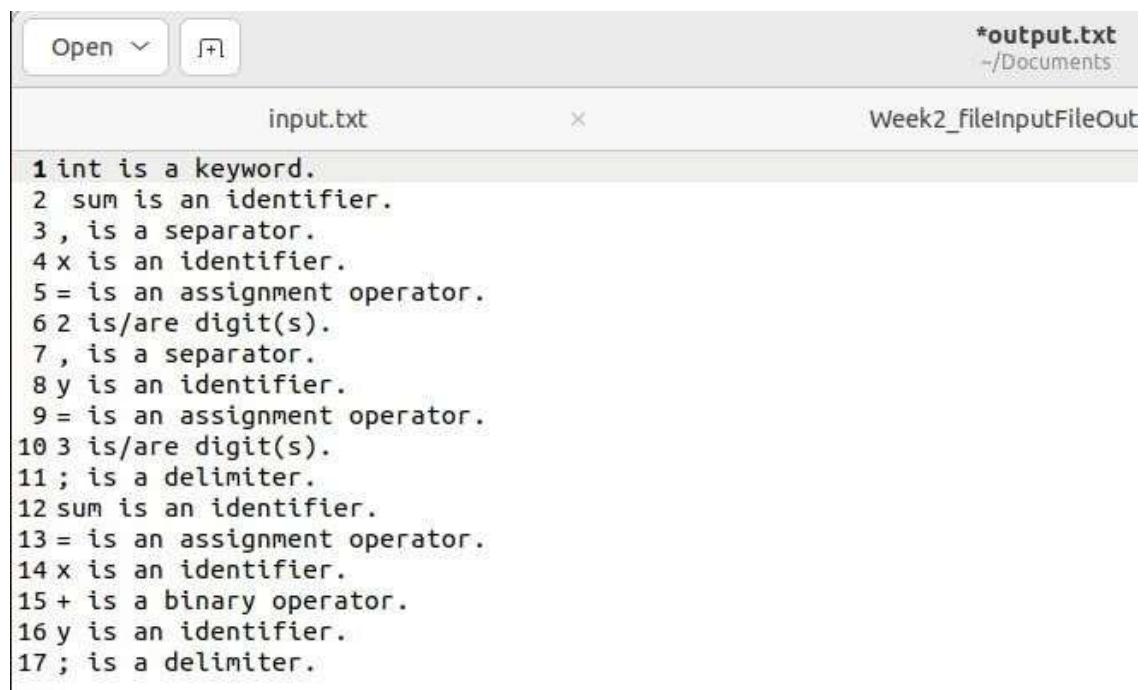


The screenshot shows a text editor window with the following details:

- File menu: Open ▾, +
- Document title: *input.txt
- File path: ~/Documents
- Content area:

```
1 int sum,x=2,y=3;
2 sum=x+y;
```
- Bottom status bar: Week2_fileInputFileOutp

Printed in output.txt



The screenshot shows a text editor window with the following details:

- File menu: Open ▾, +
- Document title: input.txt
- File path: ~/Documents
- Content area:

```
1 int is a keyword.
2 sum is an identifier.
3 , is a separator.
4 x is an identifier.
5 = is an assignment operator.
6 2 is/are digit(s).
7 , is a separator.
8 y is an identifier.
9 = is an assignment operator.
10 3 is/are digit(s).
11 ; is a delimiter.
12 sum is an identifier.
13 = is an assignment operator.
14 x is an identifier.
15 + is a binary operator.
16 y is an identifier.
17 ; is a delimiter.
```
- Bottom status bar: Week2_fileInputFileOutp

2.5 Write a program in lex to find the length of the input string.

Code

Program to find length of string
/*
 #include <stdio.h>
 int count = 0;
 Y . y
 Y .. .
 [a-zA-Z0-9.,!/?\t]+
 { length of input is %d,
 yyval); }
 */ . . .

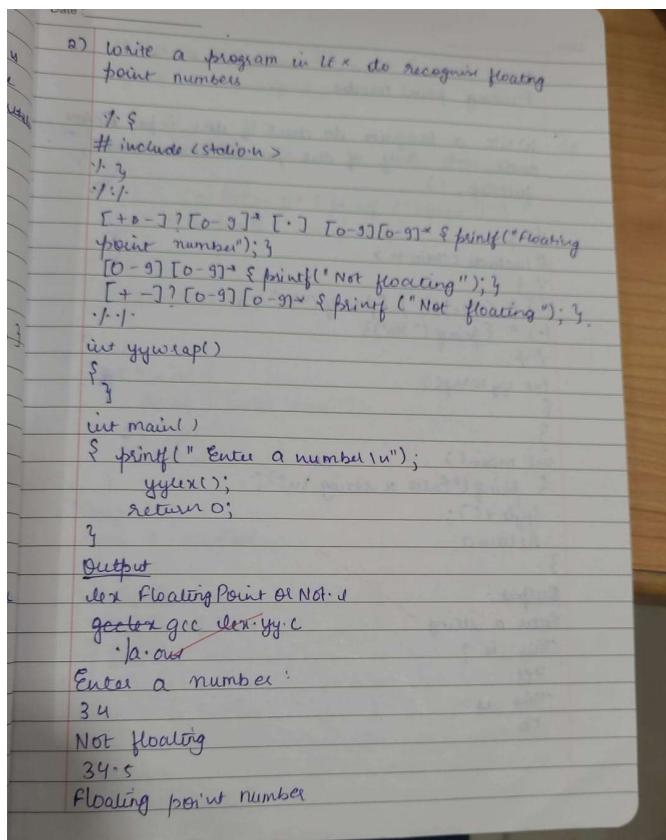
Output

```
neha29@neha-VirtualBox:~/Documents$ lex Week2_lengthOfString.l  
neha29@neha-VirtualBox:~/Documents$ gcc lex.yy.c  
neha29@neha-VirtualBox:~/Documents$ ./a.out  
Enter a string:  
Good Morning!  
Length of input string is 13.  
  
Where do you stay?  
Length of input string is 18.
```

Lab 3

3.1 Write a program in LEX to recognize Floating Point Numbers.

Code



2) Write a program in LEX to recognize floating point numbers

```
Y. $  
# include <stdio.h>  
. Y  
. Y:  
[+ -]? [0-9]* [. ] [0-9][0-9]* & printf("floating  
point number");  
[0-9][0-9]* & printf("Not floating");  
[+ -]? [0-9][0-9]* & printf("Not floating");  
. Y:  
int yywrap()  
{  
}  
int main()  
{  
    printf("Enter a number\n");  
    yylex();  
    return 0;  
}
```

Output

lex Floating Point Or Not.
gcc lex.y.c
.a.out

Enter a number:
34
Not floating
34.5
Floating point number

Output

```
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!
```

3.2 Read and 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.

Code

```
#include <stdio.h>
int flag = 0;
and [or|but|because] if [then|nevertheless] { flag = 1; }
;
In { if (flag == 1)
    printf ("yes, compound!");
else
    printf ("Simple");
}
int yywrap()
{
    return 0;
}
```

Output

Enter a string
we either go out or
stay home and
watch movie.
Compound
I am Ravjile
Simple

Output

```
neha29@neha-VirtualBox:~/Documents$ lex Week3_compoundOrSimple.l
neha29@neha-VirtualBox:~/Documents$ gcc lex.yy.c
neha29@neha-VirtualBox:~/Documents$ ./a.out
Enter a sentence:
This is a car.
Simple sentence!
neha29@neha-VirtualBox:~/Documents$ gcc lex.yy.c
neha29@neha-VirtualBox:~/Documents$ ./a.out
Enter a sentence:
She is good at singing and dancing.
Compound sentence!
neha29@neha-VirtualBox:~/Documents$
```

3.3 Write a program to check if the input sentence ends with any of the following punctuation marks (?, fullstop , !)

Code

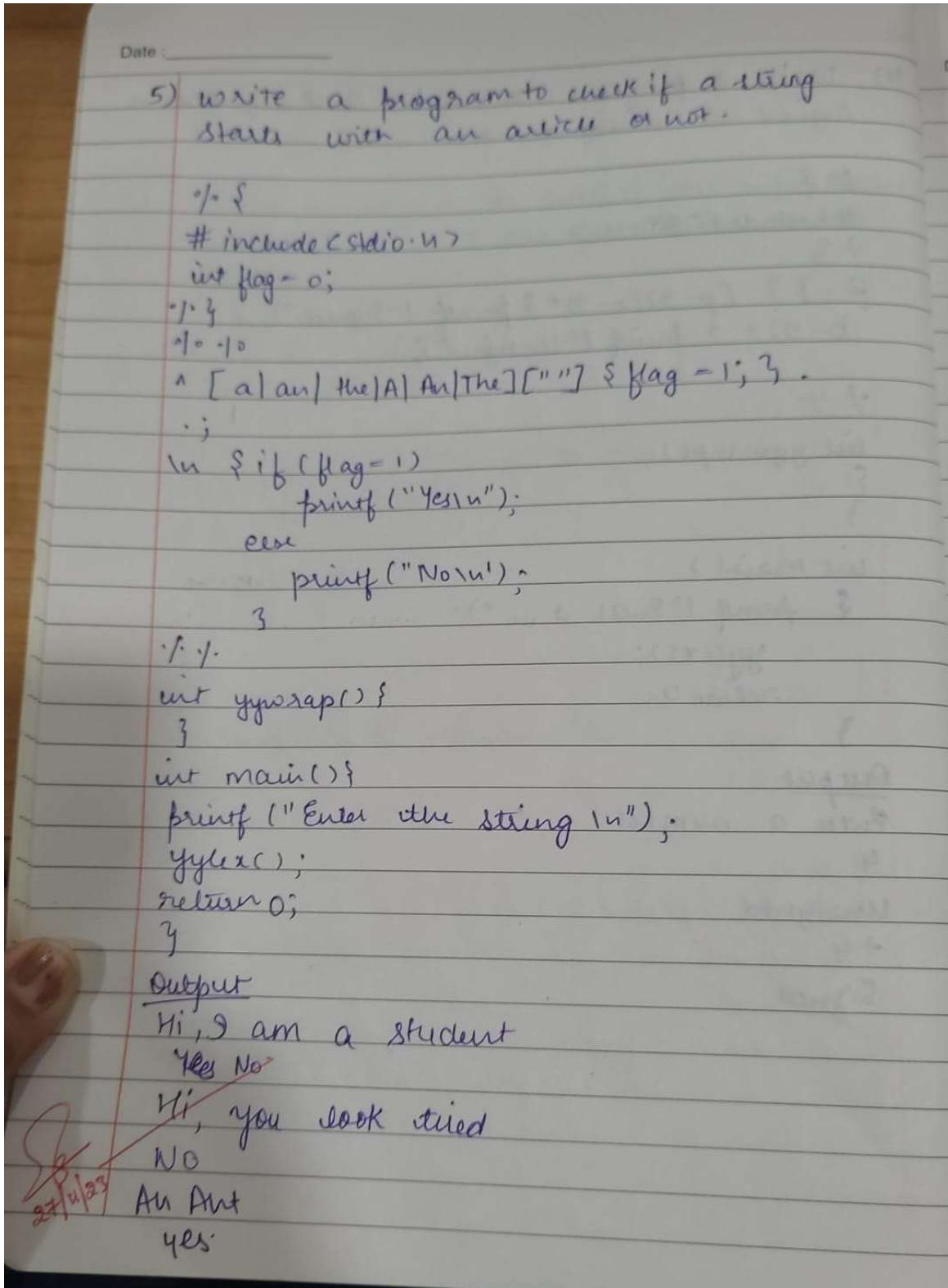
```
fullstop, ?  
{  
    #include <stdio.h>  
    {  
        if [?|.!] { printf("Yes"); }  
        else { printf("No"); }  
    }  
    int yywrap()  
    {  
    }  
int main()  
{  
    printf("Enter a string\n");  
    gets(s);  
    return 0;  
}  
  
Output:  
Enter a string  
This is ?  
Yes  
This is  
No
```

Output

```
Enter a sentence:  
Is this yours?  
Ends with a punctuation!
```

3.4 Write a program to read an input sentence and to check if the sentence begins with English articles (A, a,AN,An,THE and The).

Code



Output

Enter a sentence:

An apple a day keeps the doctor away.

Starts with an article!

3.5 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 and print the count in a file called output.txt.

Code

```
%{

#include<stdio.h>

int c=0;

%}

%%

"\\/*[^*]*\*+([/*][^*]*\*+)*\// {c++;}

"//.* {c++;}

.ECHO;

%%

int yywrap()
{
    return 1;
}

void main()
{
    yyin=fopen("input.txt","r");
    yyout=fopen("output.txt","w");
    yylex();
    printf("The number of comments are:%d\n",c);
    fclose(yyin);
    fclose(yyout);
}
```

Output

```
Enter a sentence:
//This is a comment.
No of comment lines are: 1
/*This is multi*/ //This is single.
    No of comment lines are: 2
There are no comments.
There are no comments.No of comment lines are: 0
```

3.6 Write a program to read and check if the user entered number is signed or unsigned using appropriate meta character.

Code

Date : _____

4) Write a program to read & check if a no. is signed or unsigned.

```
#include <stdio.h>
int ywzrap()
{
    int main()
    {
        printf ("Enter a no.");
        ywzrap();
        return 0;
    }
}
```

Output

```
Enter a number:
123
Unsigned number!

-123
Signed number!

+123
Signed number!
```

Lab 4

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

Code

Date: 11/12/23

Lab 4

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

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
char string[200];
```

if {

```
[ \n] { fprintf(yyout, ".\s\n", string); string[0]=10; }
[\t]* { fprintf(yyout, ".\s", string); string[0]=10;
        fprintf(yyout, ".\s", ""); }
strcat(string, yytext);
<EOF> { fprintf(yyout, ".\s", string); return; }
}
```

int main()

```
{ printf("Enter filename");
  scanf(".\s", filename);
  yyin=fopen(filename, "r");
  if (yyin == NULL)
    { perror("invalid");
      exit(0);
    }
  printf("Enter Outputfile (+)");
  scanf(".\s", filename);
```

8/11/23 - 4

Output

A screenshot of a terminal window titled "*text.txt". The window contains two lines of text: "1 Hello World" and "2 Welcome to programming|". Below the terminal window, a dark bar displays the text "Printed!".

A screenshot of a file explorer interface. On the left, there are buttons for "Open" and a plus sign. On the right, a file named "print.txt" is listed under the path "~/Documents". The file's content is shown below it: "1 Hello World" and "2 Welcome to programming".

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

4.2.1 The set of all string ending in 00.

Code

The image shows handwritten LEX code on lined paper. The code is as follows:

```
%option noyywrap
%{
#include <stdio.h>
%}

%00
[0-9]* [00] { printf("Accepted"); }
[0-9]* { printf("Rejected"); }
```

Output

```
neha29@neha-VirtualBox:~/Documents$ lex Week4_EndsWith0.l
neha29@neha-VirtualBox:~/Documents$ gcc lex.yy.c
neha29@neha-VirtualBox:~/Documents$ ./a.out
Enter a string:
12300
Ends with 0.
neha29@neha-VirtualBox:~/Documents$ gcc lex.yy.c
neha29@neha-VirtualBox:~/Documents$ ./a.out
Enter a string:
145
Does not end with 0.
neha29@neha-VirtualBox:~/Documents$
```

4.2.2 The set of all strings with three consecutive 222's.

Code

3) consecutive
Ending with 222

```
% /*  
 * [0-9]* [222] [0-9]* { printf("Consecutive 222\n"); }  
 * [0-9]* [222] [0-9]* { printf("NO consecutive 222\n"); }  
 */  
  
int main()  
{  
    yylex();  
    return 0;  
}
```

Output

```
neha29@neha-VirtualBox:~/Documents$ lex Week4_3Consecutive2s.l  
neha29@neha-VirtualBox:~/Documents$ gcc lex.yy.c  
neha29@neha-VirtualBox:~/Documents$ ./a.out  
Enter a string:  
2322  
Does not have 3 consecutive 2's.  
neha29@neha-VirtualBox:~/Documents$ gcc lex.yy.c  
neha29@neha-VirtualBox:~/Documents$ ./a.out  
Enter a string:  
322221  
Has 3 consecutive 2's.  
neha29@neha-VirtualBox:~/Documents$
```

4.2.3 The set of all string such that every block of five consecutive symbols contains at least two 5's.

Code

at least five consecutive symbols

```
int count = 0, flag
S, 5 } {flag = 0;
for(i=0 ; i < S; i++)
    S int c = yytext[i] - '0';
    if (c == 25)
        { count++;
        if (count == 2)
            { flag = 1;
            break;
        }
    }
}
```

Output

```
Enter a string:  
1525558566  
yytext:15255,flag(1 if no of 5 is atleast 2):1  
yytext:8566,flag(1 if no of 5 is atleast 2):1  
Valid string.
```

```
Enter a string:  
5432512345  
yytext:54325,flag(1 if no of 5 is atleast 2):1  
yytext:12345,flag(1 if no of 5 is atleast 2):0  
Not a valid string!
```

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

Code

Date : _____

6) The set of all strings beginning with a 1 which interpreted as binary representation is congruent to zero modulo 5

```
0/0/0
1 [01] * { for ( i = yyeng - 1; i >= 0; i - - )
    { value = value + ( yytext [ i ] - 48 ) * pow ( 2, i );
        j ++ ;
        if ( value % 5 == 0 )
            flag = 1;
    }
    [ \n ] return 0;
} /0/0
int yyseap () { }
int main ()
{ yylex ();
    if ( flag == 1 )
        printf ("success\n");
    else
        printf ("fail\n");
    return 0;
}
```

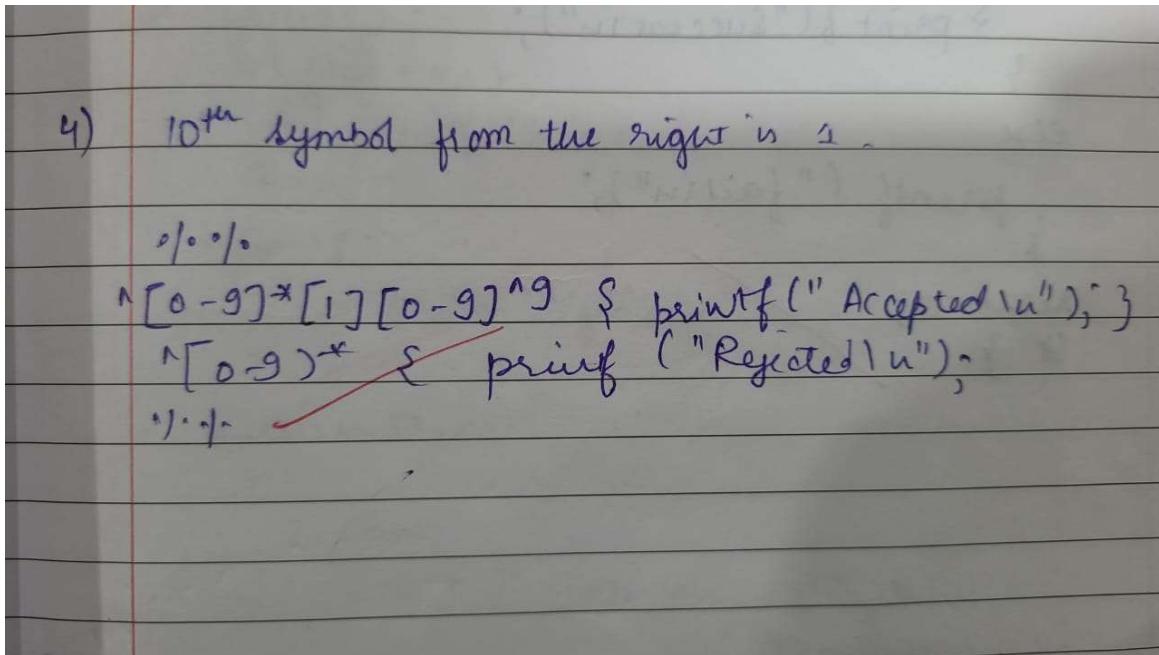
Output

```
Enter a string:  
1010  
Decimal representation:10  
Congruent to modulo 5.
```

```
Enter a string:  
111  
Decimal representation:7  
Not congruent to modulo 5.
```

4.2.5 The set of all strings such that the 10th symbol from the right end is 1.

Code



Output

```
Enter a string:  
11234345236  
10th symbol from right is 1.
```

```
Enter a string:  
23123456123  
10th symbol from right is not 1.
```

4.2.6 The set of all four digits numbers whose sum is 9.

Code

Date : _____

5) Set of all four digit whose sum is 9.
digits [0-9]
if
{
 digit4 } { digit3 } { digit2 } { digit1 } { for(i=yyleng-1;
 i>=0;i--) {
 value += (yytext[i]-48);
 }
 if (value == 9)
 { flag=1;
 }
 } n return 0;
}

int main() {
 yytext();
 if (flag == 1)
 { printf("Success\n");
 }
 else
 { printf("fail\n");
 }
 return 0;
}

Output

```
Enter a string:  
3331  
The sum of digits is not 9.  
Enter a string:  
2340  
The sum of digits is 9.
```

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

Code

Date : _____

7) Set of all four digits, whose individual digits are in ascending order from left to right
digits [0-9]

{
 {digit3} {digit2} {digit4} {digit3} if(i=0; i<yytext;
 if (yytext[i] > yytext[i+1])
 {flag=0;
 }
 }
 [1n] return;

int main()
{
 yytext();
 if (flag == 1)
 printf ("Success 1n");
 else
 printf ("Fail 1n");
 return 0;
}

21/12/2023

Output

```
Enter a string:
```

```
1235
```

```
The digits are in ascending order.
```

```
Enter a string:
```

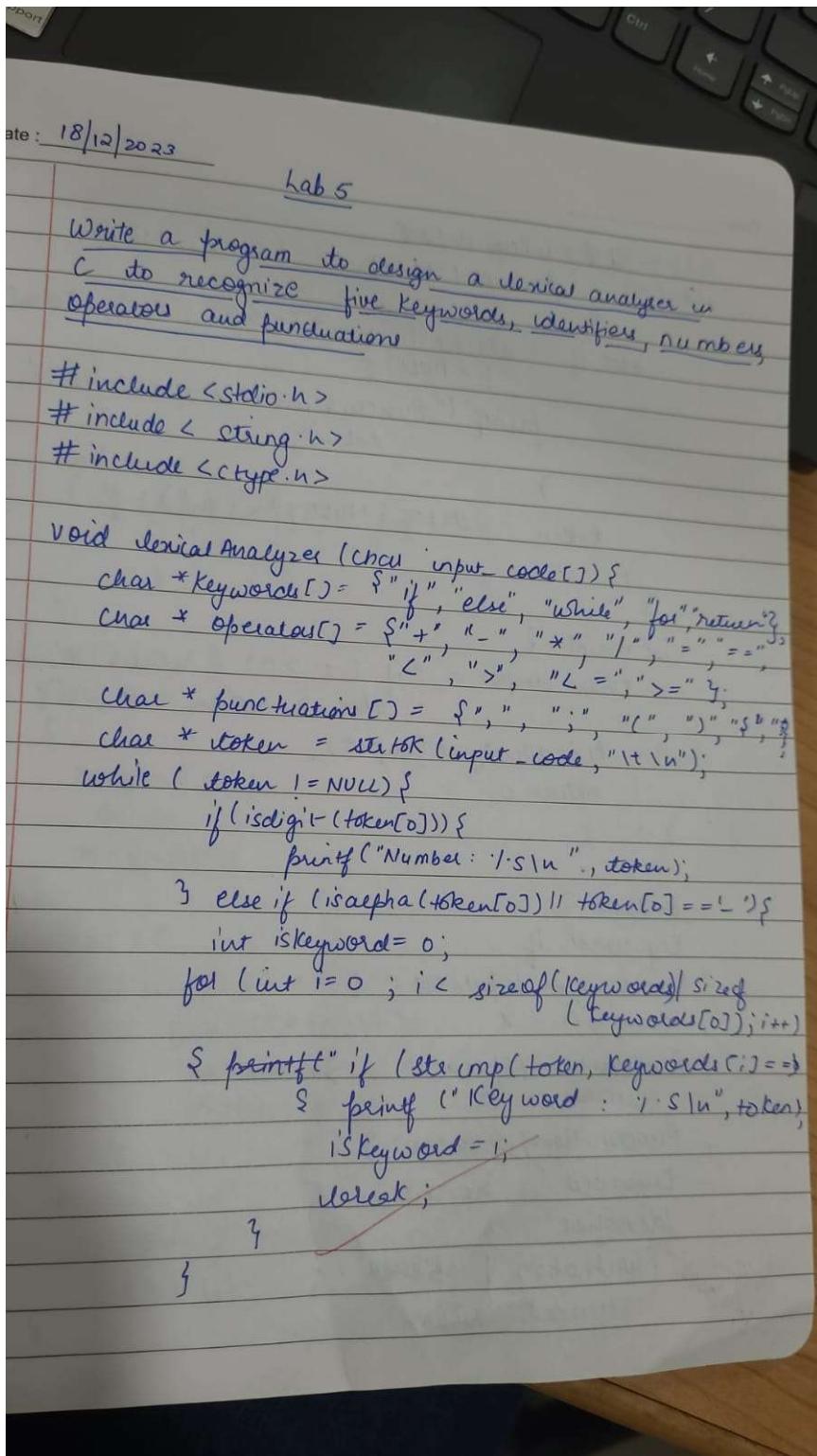
```
1243
```

```
The digits are not in ascending order.
```

Lab 5

Write a C program to design lexical analysis to recognize any five keywords, identifiers, numbers, operators and punctuations.

Code



```

Date : _____
Data : _____
if (!isKeyword) {
    printf ("Identifier : %s\n", token);
}
else if (strchr ("+-*/=<>();", token[0])
         != NULL) {
    printf ("Punctuation / operation : %s\n",
            token);
}
token = strtok (NULL, "\n\t");
}

int main() {
    char input = "if (x > 0) {return x; }
                  else { return -x; };";
    lexicalAnalyzer (input);
    return 0;
}

```

Output

```

Operator: (
Identifier: x
Operator: >
Number: 0
Operator: )
Operator: {
Keyword: return
Identifier: x
Punctuation: ;
Operator: }
Keyword: else
Operator: {
Keyword: return
Operator: -x
Punctuation: ;
Operator: }

```

Lab 6

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

S->cAd

A->ab | a

Code

Date: 8/1/24

Lab 6
Implementing Recursive Descent Parser for $S \rightarrow cAd$,
 $A \rightarrow ab/a$

```
#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 AC()
{
    if (input[ind] == 'a')
    {
        printf("Hello\n");
        match('a');
        match('b');
    }
    else
    {
        printf("failed");
        exit(1);
    }
}
```

Date : _____

```
int main() {
    printf ("Enter input string \n");
    scanf ("%s", input);
    S();
    if (input [ind] == '$') {
        printf ("Parsing Successful.\n");
    } else {
        printf ("Parsing failed. Extra character found");
    }
    return 0;
}
```

Output

```
Enter a string:
cad$
Valid string!
```

```
Enter a string:
caad$
Invalid String!
```

Lab 7

7.1 Write a program in YACC to design a suitable grammar for evaluation of arithmetic expression having +, -, * and /.

Code

Date : 8/11/2024 Lab 7
YACC

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

P.1

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

int yywrap()
{
}

P.2

%S
#include <stdio.h>
%t
%y token NUM
%y left '+'
%y right '-'
%y %y
```

Date: _____

```
exp : e & printf("Valid expression\n");  
      printf("Result : %d\n", $);  
      return 0; }  
e: e+'e' { $ = $1 + $3; }  
| e-'e' { $ = $1 - $3; }  
| NUM { $ = $1; };  
T: .  
int main(){  
    printf("Enter arithmetic expression\n");  
    yyparse();  
    return 0;  
}
```

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

OUTPUT

See
8/1/20

Enter an arithmetic expression

5+6-3-6

Valid expression

Result: 14.

Enter an arithmetic Expression

5-6 -

Invalid expression

Output

```
Enter an arithmetic expression:
```

```
2+3*4
```

```
Valid expression!
```

```
Result:14
```

7.2 Write a program in YACC to recognize strings of the form $\{(a^n)b, n \geq 5\}$.

Code

Date : _____

String Matching : $a^n b$

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

P.y

```
'.' $ #include <stdio.h>  
int yyerror(char *s);  
int yylex(void);  
'.' .  
. token A  
. token B  
. token NL  
. .  
sent : A AA AA S B NL { print ("Passed using  
rule b^n),  
n >= 5 in
```

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

Output

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

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

Code

Date : 29/1/24

Write a yacc program to generate syntax tree
for the given arithmetic expression

```
P·d
Y·s
# include "y.tab.h"
extern int yyval,
    y;
[0-9]* {yyval = atoi(yytext);
    return digit;}
[+];
[=] return 0;
    return yytext[0];
}
int yywrap()
{
    return 1;
}
```

P·y

#include
include <stdio.h>
include <math.h>
include <stdlib.h>
include <string.h>

struct tree_node
 { char val[10];
 int lc;
 int rc;
 };
 int cur_id;
 struct tree_node syn_tree[100];
 void my_point_tree(int cur_id);
 int mk_node(int lc, int rc, char val[10]);
 /*
 * token digit
 */
 S: E { my_point_tree(\$1); }
 ;
 E: E (+) T { \$ \$ = mk_node(\$1, \$3, "+"); }
 / T { \$ \$ \$ = \$1; }
 ;
 T: T (*) F { \$ \$ \$ = mk_node(\$1, \$3, "*"); }
 / F { \$ \$ \$ = \$1; }
 ;
 F: '(' E ')' { \$ \$ \$ = \$2; }
 if digit { char buf[10]; sprintf(buf, "%d",
 val);
 \$ \$ \$ = mk_node(-1, -1, buf); }
 /* */

```

date
int main() {
    int id = 0;
    printf("Enter an expression");
    yyparse();
    return 0;
}

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

int mknod (int id, int xc, char val[10])
{
    strcpy (syn-tree[id].var, val);
    syn-tree [id].lc = -1;
    syn-tree [id].rc = -1;
    id++;
    return id - 1;
}

void my-print-tree(int cur_id)
{
    if (cur_id == -1) return;
    if (syn-tree[cur_id].lc == -1 & syn-tree
        [cur_id].rc == -1)
        printf ("Digit Node-> Index: %d, value '%c',
               left-child Index : %d, Right child Index:
               %d.\n", cur_id, syn-tree [cur_id].val,
               syn-tree [cur_id].lc, syn-tree [cur_id].rc);
    my-print-tree (syn-tree [cur_id].lc);
    my-print-tree (syn-tree [cur_id].rc);
}

```

Output

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

```

Lab 8

8.1 Write a program in YACC to convert infix to postfix expression.

Code

Date : 29/1/24 Lab 8

p. 1

```
y. $  
# include "ytab.h"  
extern int yyval;  
y. }  
y. y.  
[0-9]+ { yyval = atoi (yytext); return digit; }  
[t];  
[n]; return 0;  
return yytext[0];  
y. y.  
int yyparse()  
S  
y
```

p. 2

```
y. $  
# include <ctype.h>  
# include <stdio.h>  
# include <stdlib.h>  
y. }  
y. return digit  
y. y.
```

te : _____

S: E { printf("\n\n"); }

;

E: E '+' T { printf(" + "); }

| T

;

T: T '*' F { printf('*'); } | F

;

F: 'C' E 'I'

| digit { printf(".%d", \$1); }

;

T . I.

int main () {

printf ("Enter infix expression");

yytoken();

}

yyerror()

{

printf("Error");

}

Output:

Enter infix expression 2 + 6 * 3 + 4
2 6 3 * + 4 +

Output

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

Lab 9

9.1 Write a program in YACC to generate three address code for a given expression.

Code

Date : 29/1/24 Week 9

Three Address Code

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

d[0-9] +
a [a-zA-Z] +
% y.

S d { yyval = atoi(yytext); return digit; }
{ a y { strcpy(yytext, yytext); yyval=1; return id; }
( t ) { ; }

in return;
return yytext[0];
' y.
int yyparse()
{
    return 1;
}

P.y
```

% S

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

Date : _____

```

int yyparse(char *s);
int yyparse(void);
int val_int = 0;
char token [20];
}

/* token id
/* token digit
*/
S: id '=' E {printf (" -> s=t+d\n", iden, var_cnt)};
E: E + T {var_cnt++; var_cnt++, printf (" + d =
    t+d + t+d; \n"), $1, $2, $3, $4};
T: T * F {var_cnt++, var_cnt++, printf (" * d =
    t+d * t+d + t+d\n"), $1, $2, $3, $4};
F: P | N | F {var_cnt++; var_cnt++, printf (" t+d =
    t+d + t+d; \n"), $1, $2, $3, $4};
P: '(' E ')' {printf (" -> s=d\n"), $1, $2, $3, $4};
L digit {var_cnt++; var_cnt++, printf (" t+d\n"),
    $1, $2, $3, $4};
}

```

Date : _____

```

(" + d = d; \n", $1, $2, $3, $4);
};

/* main()
int main(){
    var_int = 0;
    printf (" Enter an expression: \n");
    yyparse();
    return 0;
}

int yyparse(char *s)
{
    if (printf (" Invalid expression! \n"));
    return 0;
}

```

Output

Output

```
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
```