

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT

on

COMPILER DESIGN

Submitted by

Kshitij S(1BM21CS093)

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 **Kshitij(1BM21CS093)**, 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. 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, Kshitij S(1BM21CS093), 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		6-7
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 vowels and consonants in a string.	7
2		8-15
2.1	Write a program in lex to count the number of words in a sentence.	8
2.2	Write a program in lex to demonstrate regular definition.	9
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.	10-11
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.	12
3		14-22
3.1	Write a program in LEX to recognize Floating Point Numbers.	14-15
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.	16-17
3.3	Write a program to check if the input sentence ends with any of the following punctuation marks (?, fullstop , !)	18
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).	19
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.	20-21
3.6	Write a program to read and check if the user entered number is signed or unsigned using appropriate meta character.	22
4		23-33
4.1	Write a LEX program that copies a file, replacing each nonempty sequence of white spaces by a single blank.	23-24
4.2	Write a LEX program to recognize the following tokens over the alphabets {0,1,..,9}	25

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.	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
4.2.5	The set of all strings such that the 10th symbol from the right end is 1.	31
4.2.6	The set of all four digits numbers whose sum is 9.	32
4.2.7	The set of all four digital numbers, whose individual digits are in ascending order from left to right.	33
5		35
5.1	Write a C program to design lexical analysis to recognize any five keywords, identifiers, numbers, operators and punctuations.	35-37
6		38-40
6.1	Write a program to perform recursive descent parsing on the following grammar: S->cAd A->ab a	39-40
7		41-43
7.1	Write a program in YACC to design a suitable grammar for evaluation of arithmetic expression having +, -, * and /.	41-43
7.2	Write a program in YACC to recognize strings of the form $\{(a^n)b, n \geq 5\}$.	44-45
7.3	Write a program in YACC to generate a syntax tree for a given arithmetic expression.	46-49
8		50-51
8.1	Write a program in YACC to convert infix to postfix expression.	50-51
9		52-55
9.1	Write a program in YACC to generate three address code for a given expression.	52-55

Lab 1

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

Code:

1) To identify keywords, identifiers in a program

Y. option noyywrap
~1. {
#include <stdio.h>
X }
Y. ~.
int | float | char | double ("keywords"
[a-zA-Z]* | double ("Identifiers"
, | ; | double ("separators")));

Output

```
Give an input:  
int sum,x=2,y=3,z;  
int-keyword  
sum-Identifier  
,-separator  
x-Identifier  
=-assignment operator  
2-digit  
,-separator  
y-Identifier  
=-assignment operator  
3-digit  
,-separator  
z-Identifier  
;-delimiter
```

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

Code

Count no of Vowels & Consonant
i. option noyywsh
ii. q
#include <stdio.h>
i.
Y.Y.
alelilololu | A | E | I | O | U { b++
[a-zA-Z] { c++ ; }
In i found ("number of vowel
number of consonant
Y.Y.

Output

```
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
```

Lab 2

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

Code

4. Count No of words in Input
1. Option no ywsh
2. #include <stdio.h>
int c=0;
3.

Output

```
Enter a sentence:  
This is compiler design lab work  
No of words in the sentence  
The sun rises in the east and se
```

2.2 Write a program in lex to demonstrate regular definition.

Code

Q3 write a LEX program to identify character and numbers
[0-9]+ {printf ("%.s is digit\n", s)}
[a-zA-Z]* {printf ("%.s is char\n", s)}

Output
123
123 is digit
sum

Output

```
Enter a string:  
HelloWorld  
Characters  
  
1234  
Digits  
Hello123  
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

```
o Read input from a file and print tokens  
1. option noyywrap  
1. h  
    includ c stdio.h  
1. {  
1. }  
[0-9]*          { float ("") ;  
[a-zA-Z]*        { printf ("\"%s\"");  
[a-zA-Z0-9]*    { printf ("\"%s\"");  
1. }  
void main()  
{  
    printf ("Enter file name: ");  
    scanf ("%s", fname);  
    yypin = fopen (fname,  
    yyflex (?);  
    yyin = yyin; }
```

Output

The screenshot shows a terminal window with the following details:

- File menu: Open, New
- Current file: *input.txt
- Working directory: ~/Documents
- Output window title: Week2_fileInputFileOutp
- Content of the output window:

```
int is a keyword.  
sum is an identifier.  
, is a separator.  
x is an identifier.  
= is an assignment operator.  
2 is/are digit(s).  
, is a separator.  
y is an identifier.  
= is an assignment operator.  
3 is/are digit(s).  
; is a delimiter.  
sum 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.

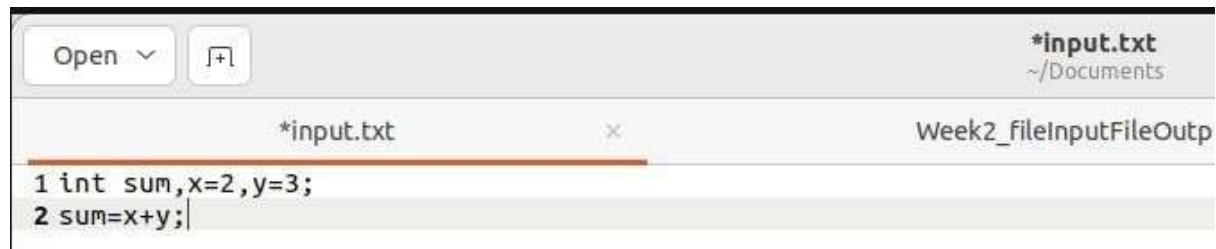
Code

Q8 Modify above code write
1. option noyywrk
2.
#include < stdio.h >

1. 2.
[0-9]* different (yyout, " "
[a-zA-Z]* different (yyout, " "
[0-9a-zA-Z]* different (yyout, " "

Y. Y.
void main()
{ char fname[20], fname;
printf (" Enter file p: ");
scanf (" -s ", fname);
printf (" Enter outfile
scat (" -s ", fname) ;
yyin = fopen (fname, "r");
yyout = fopen (fname, "w");
yylexc ();
fscanf (yyin);
fscanf (yyout);
}

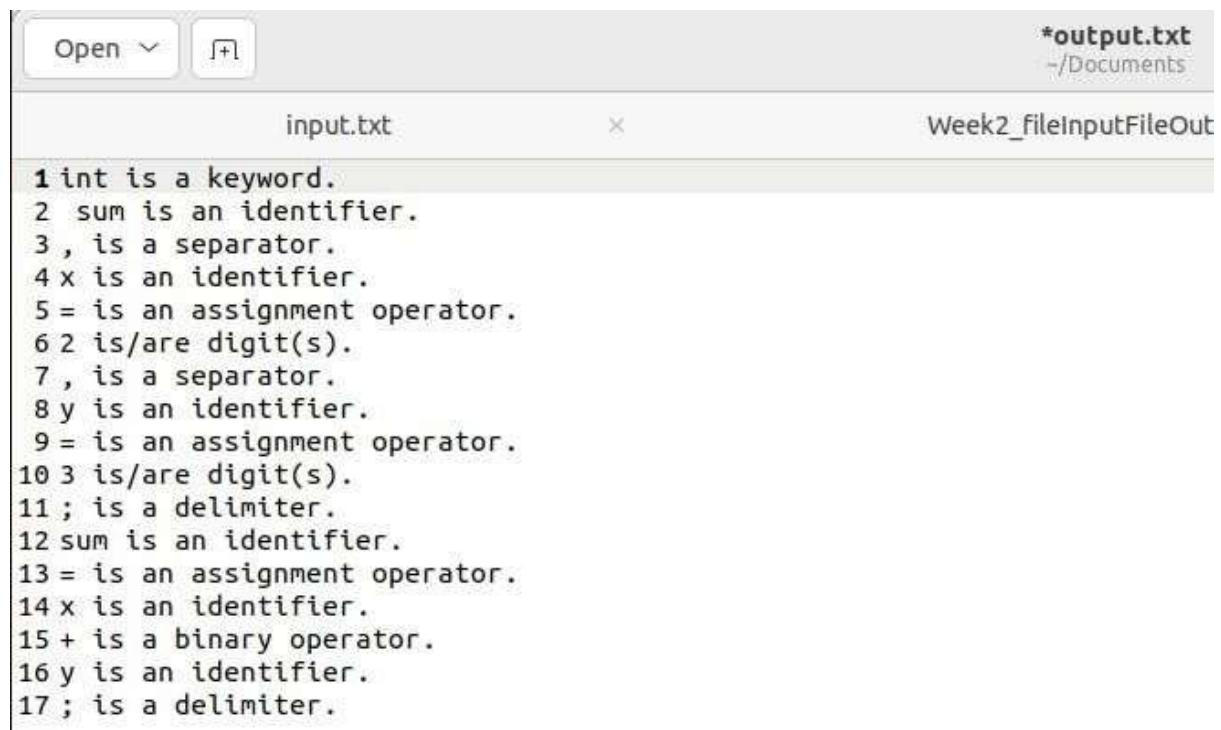
Output



A screenshot of a text editor window. The title bar shows the path `~/Documents` and the file name `*input.txt`. The main area contains the following code:

```
1 int sum,x=2,y=3;
2 sum=x+y;
```

Printed in output.txt



A screenshot of a text editor window. The title bar shows the path `~/Documents` and the file name `*output.txt`. The main area contains the following text, which appears to be the output of a lexical analyzer:

```
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.
```

Lab 3

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

Code

Lab Program 1

a. write a program in LEX to recognize Floating Point Numbers. Check for all invalid characters.

```
% option noyywrap  
%  
#include <stdio.h>  
%  
%  
[-+]?[0-9]*[.]?[0-9]*  
[-+]?[0-9]* found ("Integ"  
%  
int main()  
{  
    yylex();  
    return 0;  
}  
Output  
a -> lex & lab1.l
```

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

Practice Question

Q Check if a sentence is compound
This compound if sentence has and ,
or ,but ,because ,if ,then ,nevertheless

1. option newword
1. d
include <stdio.h>
int a=0
{
 }
-1.-1.
and | or | but | because | if | then |
[a-zA-Z]* d ?
In { return 0; }
-1.-1.
int main()
{
 printf ("Enter the input:
 gets();
 if (a == 1)
 printf ("It is compound
 }
}

Output

```
Enter a sentence:  
This is a car.  
Simple sentence!
```

```
Enter a sentence:  
She is good at singing and dancing.  
Compound sentence!
```

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

Code

The image shows handwritten notes on lined paper. At the top, there is a question mark followed by the text "Check if input sentence ends". Below this, there is a large crossed-out section containing the text "[?!.!:] \$". To the right of this, there is some illegible handwriting. Underneath the crossed-out section, there is more handwriting: "[a-zA-Z]* d \n" and "In \n return 0;". Below this, there is another question mark followed by the word "Output". At the bottom, there is a line of text starting with "ad. Enter the input sent".

Output

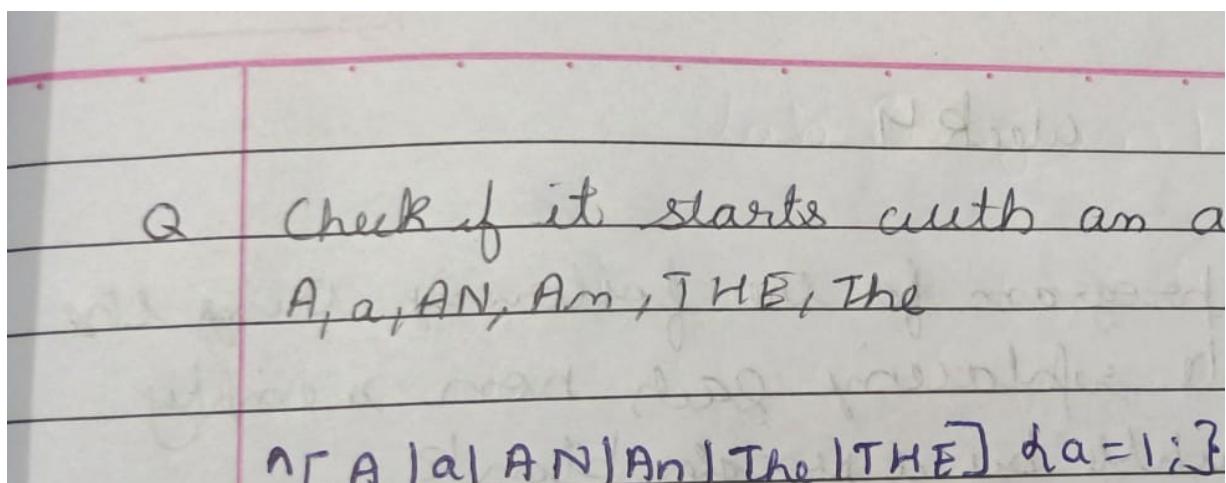
```
Enter a sentence:  
Is this yours?  
Ends with a punctuation!
```

```
Enter a sentence:  
Amazing!  
Ends with a punctuation!
```

```
Enter a sentence:  
You are good.  
Does not end with 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:  
This is a good idea.  
Does not start with an article!  
Enter a sentence:  
Amazing experience!  
Does not start with an article!  
Enter a sentence:  
The sun rises in the east.  
Starts with an article!  
Enter a sentence:  
A book is lying on the table.  
Starts with an article!  
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

```
a program. Read the input from a file  
and print the count in a file called  
input.txt  
int c = 0;  
if (/* [^*/] * \/* + ([^/*][^*]  
"/*". * {c++;}  
.ECHO;  
/*.  
int yywrap()  
{  
    return 1;  
}  
void main()  
{  
    yyin = fopen ("input.txt", "r");  
    yyout = fopen ("output.txt", "w");  
}
```

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

Q | check of signed or un
n [+|-] { a=1; }
[0-9]* { }
In { return 0; }

Output
Enter the input: 123
Unsigned number!

Output

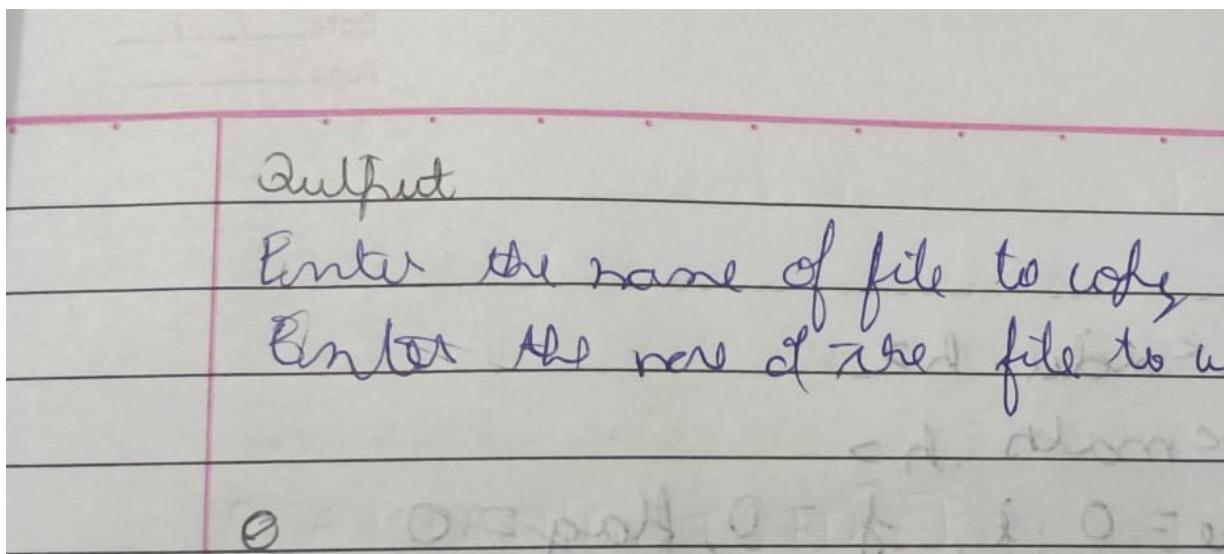
```
Enter a number:  
123  
Unsigned number!  
  
-123  
Signed number!  
  
+123  
Signed number!
```

Lab 4

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

Code

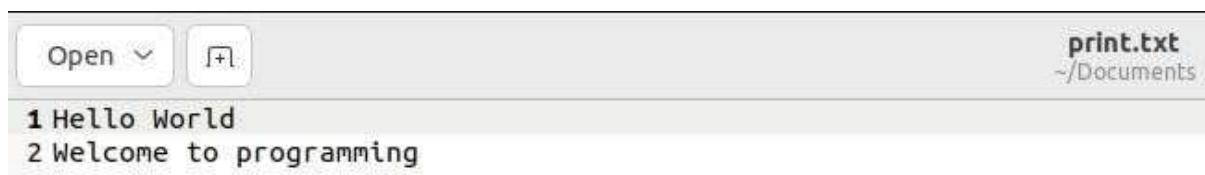
```
Q write a lex program for the fol
copies a file replacing each
sequence of white by a single b
% option noyywrap
y. d
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
char str[200];
y. y
y. y
[ \n ].fflush(yyout, "r"
str
[ ] ~ [ \t ] { fflush(yyout, "r",
str[0])
fflush(yyout, "r"
. sbrl(str, yytext);
c c EOF => fflush(yyout, "r
re
y. y
int main()
{
```



Output

```
*text.txt
1 Hello      World
2 Welcome to      programming|
```

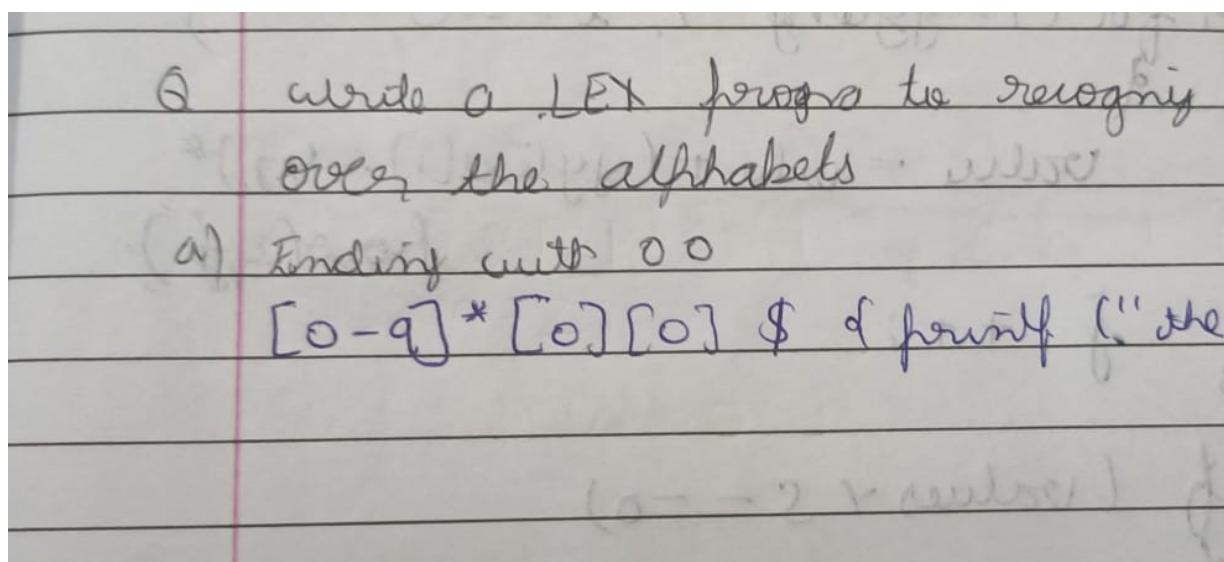
Printed!



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

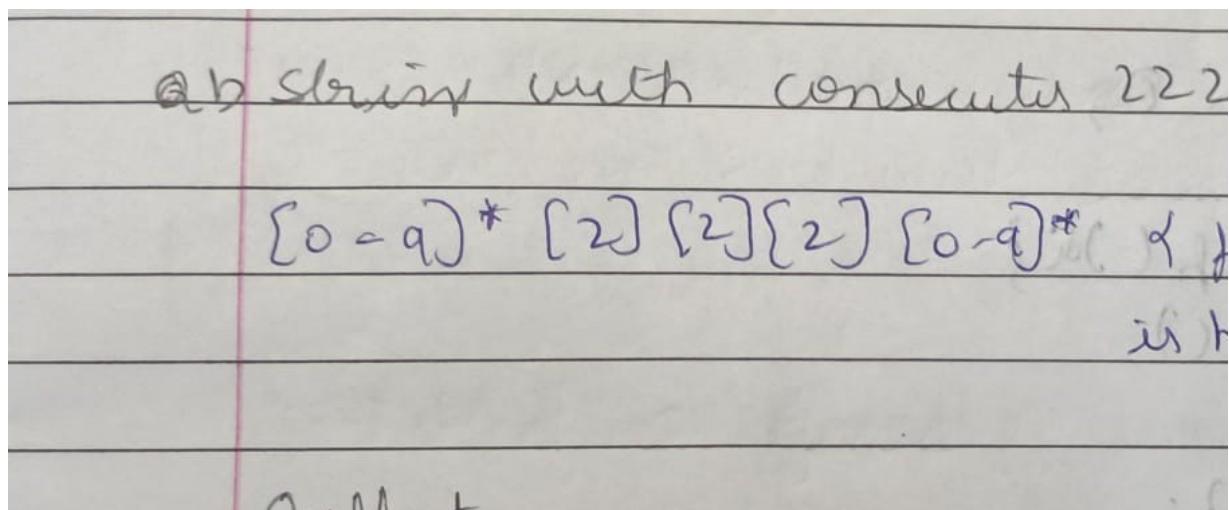


Output

```
Enter a string:  
12300  
Ends with 0.  
Enter a string:  
145  
Does not end with 0.
```

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

Code



Output

```
Enter a string:  
2322  
Does not have 3 consecutive 2's.
```

```
Enter a string:  
322221  
Has 3 consecutive 2's.
```

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

Code

Q Write a C program to recognise following alphabet

```
#include <stdio.h>
int i, cont = 0; flag;
if (cont == 2)
    flag = 1;
for (i = 0; i < 5; i++)
    if (cont == 0)
        if (c == '5')
            cont++;
    if (cont == 2)
        flag = 1;
        break;
    }
}
cont = 0
printf("Entered: %s",
```

```

void main()
{
    printf ("Enter a string : \n");
    gets (str);
    if (flag == 1)
        printf ("Valid string. \n");
    }
    int getword()
    {
        return 1;
    }
}

```

Output
5 extra. string

Output

```

Enter a string:
1525558566
yytext:15255,flag(1 if no of 5 is atleast 2):1
yytext:58566,flag(1 if no of 5 is atleast 2):1
Valid string.

Enter a string:
12345455
yytext:12345,flag(1 if no of 5 is atleast 2):0
Not a 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

```
#include <stdio.h>
#include <math.h>
int value = 0; i, j = 0, flag = 0;
y. l
    if (value % 5 == 0)
        flag = 1;
    else
        flag = 0;
    if (flag == 1)
        printf("%c", '1');
    else
        printf("%c", '0');
}
y. y
    value = value + pow(2, i);
    i++;
}
y. s
if (value % 5 == 0)
    flag = 1;
else
    flag = 0;
if (flag == 1)
    printf("%c", '1');
else
    printf("%c", '0');
}
y. r
int yywrap()
int main()
{
```

Output

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

```
Enter a string:  
101  
Decimal representation:5  
Congruent to modulo 5.
```

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

```
Enter a string:  
123  
Not a binary number.
```

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

Code

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

$[0-9]^* 1 [0-9] [0-9] \dots [0-9]$ 9 times

Output

Output

```
Enter a string:  
23123456123  
10th symbol from right is not 1.  
Enter a string:  
11234345236  
10th symbol from right is 1.
```

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

Code

The image shows handwritten C code on lined paper. A vertical red line is drawn on the left. The code starts with 'f)' followed by a comment 'The set of all 4 digits where'. Below this, there is a loop structure starting with '(0-9) [0-9] [0-9] [0-9]'. A red checkmark is placed under the first three brackets. To the right of the code, there are annotations: 'for i = yyle' with '0' and '1' above it, 'value -' with a brace below it, 'if (value' with a brace below it, and 'of' with a brace below it, followed by '- fla'.

```
f)
The set of all 4 digits where
(0-9) [0-9] [0-9] [0-9] for i = yyle
    0
    1
    value -
    }
    if (value
    {
        - fla
    }
```

Output

```
Enter a string:
6300
The sum of digits is 9.
```

```
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

```
g1 (a, b, c, d) {  
    int i, j, k, l;  
    int count = 0;  
    for (i = a; i <= d; i++) {  
        for (j = i + 1; j <= d; j++) {  
            for (k = j + 1; k <= d; k++) {  
                for (l = k + 1; l <= d; l++) {  
                    if (i < j < k < l) {  
                        count++;  
                    }  
                }  
            }  
        }  
    }  
    return count;  
}
```

Output

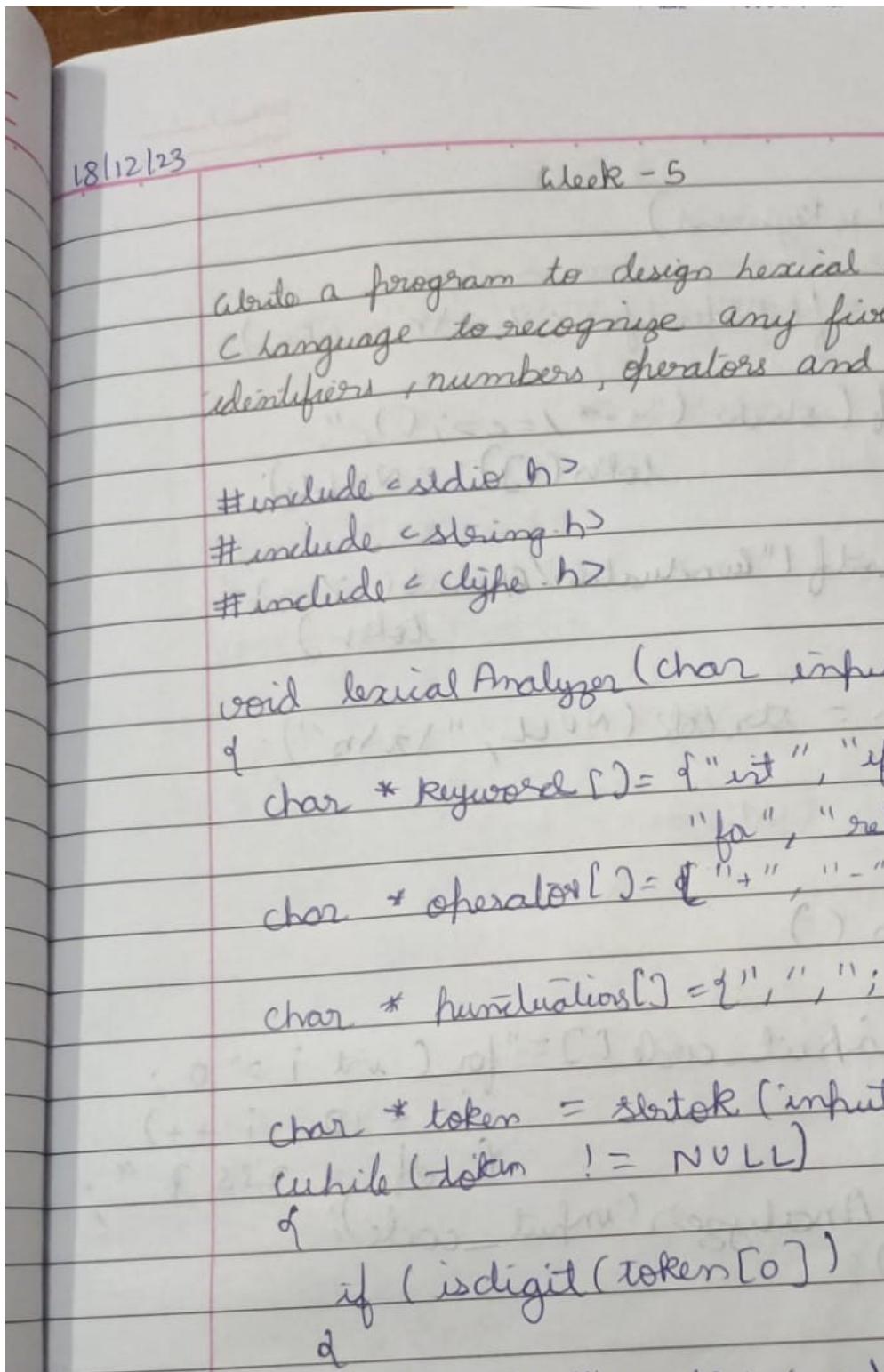
```
Enter a string:  
1235  
The digits are in ascending order.
```

```
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



if (!isKeyword)

{

 printf ("Identifier : %s\n")

}

else if (starts ("+-*/=>>=|")) {

 token[0] =

}

 printf ("Punctuation/Operator : %c\n")

}

 token = strtok (NULL, "

 {

int main ()

{

 char input_code [] = "for

}

 lexical Analyzer (input_code, 0);

Output

```
Keyword: if
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

Week 6

Write a program to perform Rec
Parsis on the following gram

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

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

```
int main()
{
    printf ("Enter the input string\n");
    scanf ("%s", input);
    sc();
    if (input [end] == '$')
    {
        printf ("Parsing successful.\n");
    }
    else
    {
        printf ("Parsing failed. Extra\n");
    }
    return 0;
}
```

Output

Enter the input string:
cad\$

Parsing successful

Output

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

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

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

Lab 7

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

Code

| Kh - lab ()

Week 7

Q Design a suitable grammar for arithmetic expression having + operators

^ highest priority &

^ more highest &

* / third " &

+ , - least " &

file1 P. L

-1.h

```
#include "y.tab.b
y;
y-y;
[0-9]+ lgyval = atoi (
[1+] ;
in      return 0;
else gyltext[0];
-      return gyltext[0];
y-y;
int gylval()
{
    }
```

expr: e + himself ("Valid expression")
printf ("Result: %d
return 0; }

e: e + t { \$\$ = \$1 + \$3 }
| e - t { \$\$ = \$1 - \$3 }
| t { \$\$ = \$1; }

*:
t: t * f { \$\$ = \$1 * \$3 }
| t / f { \$\$ = \$1 / \$3 }
| f { \$\$ = \$1; }

;
f: f * r { \$\$ = \$1 * \$3; }
| g { \$\$ = \$1; }

r: r * a { \$\$ = \$1 * \$3; }
| q { \$\$ = \$1; }
q: NUM { \$\$ = \$1; }

- f.

int main()

{

printf ("In Enter a arithmetic
expression: ");

Output

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

29/01/24 week - 7

Q Write code for string match
string match . 1

Y. f

#include < stdio.h >

#include < stdlib.h >

#include < y.tab.h >

extern int yylex

Y. ?

Y. Y.

[a A] { yylex = yylex[0]; }

[b B] { yylex = yylex[0]; }

In { return NL; }

. { return yylex[0]; }

Y. Y

int yygraph()

{

return 1;

}

String match . y

Y. f

```
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!  
aaaaaaab  
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

29/01/24

Q. write code for syntax tree g

Syntax Tree . l

```
#include <stdio.h>
#include <stdlib.h>
#include "y.tab.h"
extern int yyval;
```

Y. }

Y. }

[o - a] + t yyval = ate
return dig

[\t];

[\n] return 0;

. return yylext[o];

Y. ;

int yeyroot()

{

return 1;

}

struct tree node

{
char val[10];

int le;

int re;

};

end node

struct tree node symtree[100];

void my print tree (int cur ind
int mknodc (int le, int re

);

γ. token digit

γ. γ.

S : E { my print tree (\$1) ; }

;

E : E '+' T { \$\$ = mknodc (\$1

| T d. \$ = \$1; }

;

T : T '*' F { \$\$ = mknodc (\$1, \$

| F d. \$ = \$1; }

;

γ. γ.

int mknode (int lc, int rc, char ch)

{

 strcpy (synTree [ind],

 synTree [ind]).lc = lc;

 synTree [ind].rc = rc;

 ind++;

 return ind - 1;

}

al).

void myPrintTree (int curInd)

{

 if (curInd == 1) return;

 if (synTree [curInd].lc !=

syn

 printf ("Digit Node → Index

 curInd, syn

 else

 printf ("OperatorNode → Index

 left child index: %d

 curInd, synTree [

 synTree [curInd]

 myPrintTree (synTree [curInd])

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

29/01/24 week 8

Infix To Postfix

InfixToPostfix.l

```
#include <stdio.h>
#include <stdlib.h>
#include <y.tab.h>
extern YYSTYPE yylval;
```

Y. {
 #include <stdio.h>
 #include <stdlib.h>
 #include <"y.tab.h">
 extern YYSTYPE yylval;
}
Y. }
Y. Y.
[0-9]+ d yylval = atoi(yylval);
[+ - * /] ;
In { return 0; }
d return yytext[0]; }
Y. Y.
int yywrap()
{
}
InfixToPostfix.y

```

s: e + f
;
e: e '+' t
| e '-' t
| t
;
t: t '*' n
| t '/' n
| h
;
f: '(' e ')'
| num
;
y-y.
void main()
{
    printf("Enter an infix expression");
    yyparse();
}
int yyerror(const char *s)
{
    printf("Invalid infix expression");
}

```

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

29/01/29 week - 9

Q. Write a program in YACC to generate 3a given exp

AddressCode.l

```
Y. {  
    #include <stdio.h>  
    #include <stdlib.h>  
    #include "y.tab.h"  
    extern int yyval;  
    extern char yytext[20];  
  
Y. }  
d [0-9]+  
a [a-zA-Z]+  
Y. Y.  
{ d } i symbol = atoi (yytext)  
{ a } i strcpy (iden, yytext);  
(+) { ;  
In return 0;  
- return yytext[0];  
X. }  
int main ()
```

Address Code.y

{

-1-2

#include <math.h>

#include <ctype.h>

#include <stdio.h>

int yyparser(char *s);

int yylex(void);

int varent = 0;

char sder(20);

Y. }

Y. token id

Y. token digit

Y.-Y.

S: id '=' E { printf("Ys=")}

E: E '+' T { \$ = varent; varent++;

printf("T+1.d = %d\n")

| E '-' T { \$ = varent; varent++;

printf("T-1.d = %d\n")

IT { c \$\$ = \$1 }

;

T: T '*' T { \$ = varent; varent++;

P: 'C' E ')' { \$ = \$2 : }
1 digit { \$ = current ; over
printf (" + %d = %d : ")

;

-1 -1.

int main ()

{

current = 0;

printf ("Enter an expression
yyparse ();

return 0;

}

int yyerror (char *s)

{

printf ("Invalid Expression
return 0;

}

Output

Enter an expression

a = 2 * 3 / 6 - 4

40 = 2

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
```