

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT

on

COMPILER DESIGN

Submitted by

Abhinav Ishan(1BM21CS273)

Under the Guidance of
Prof. Sonika S D
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 **Abhinav Ishan (1BM21CS273)**, 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. Latha NR
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, Abhinav Ishan (1BM21CS273), 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. Sonika S D, 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

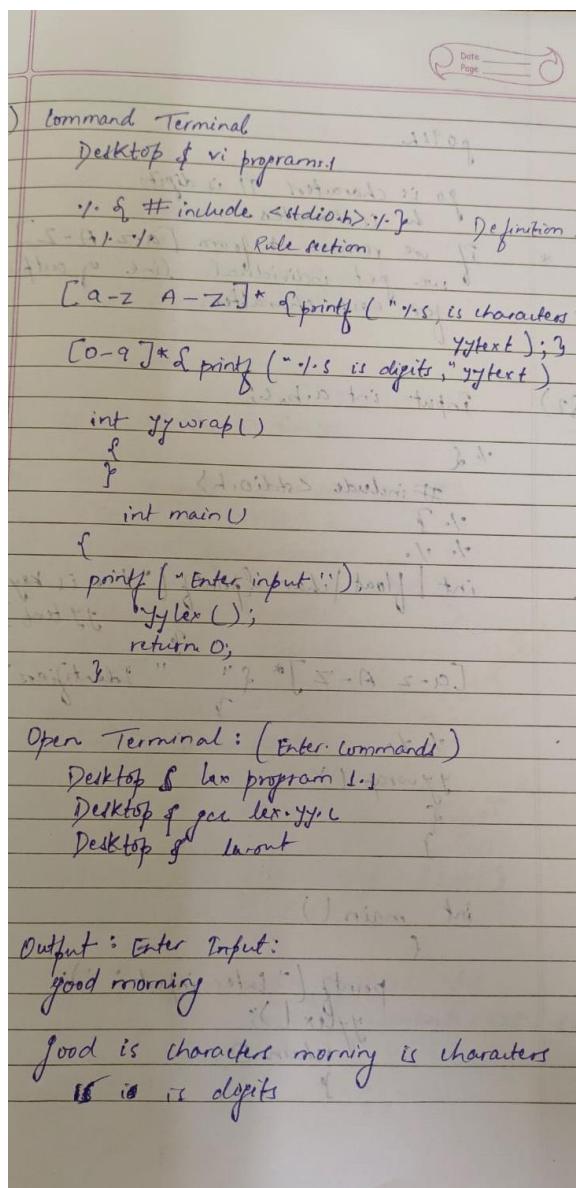
Lab No	Title	Page No
1		6-8
1.1	Write a program in LEX to recognize different tokens: Keywords, Identifiers, Constants, Operators and Punctuation symbols.	6
1.2	Write a program in LEX to count the number of characters and digits in astring.	7
	Write a program in LEX to count the number of vowels and consonants in astring.	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 afile 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 afile 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 compoundelse it is simple.	17
3.3	Write a program to check if the input sentence ends with any of the followingpunctuation marks (?, fullstop , !)	18-19
3.4	Write a program to read an input sentence and to check if the sentence beginswith 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 printthe 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:



Handwritten LEX program code:

```
command Terminal
Desktop $ vi programs.l
%{ #include <stdio.h> %} Definition
%{ %} Rule action
[a-zA-Z]* printf ("%s is characters"
[0-9]* & printf ("%s is digits", yytext)
int yywrap()
{
    // A digit is defined as
    int main()
    {
        printf ("Enter input: ");
        yylex();
        return 0;
    }
}

Open Terminal : (Enter commands)
Desktop $ lex programs.l
Desktop $ gcc lex.yy.c
Desktop $ ./lex
```

Output: Enter Input:

good morning

good is characters morning is characters

15 is digits

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 characters and digits in a string.

Code

Output length of given input string
Date _____
Page _____

int a,b,c;
int is keyword
a is keyword identifier separator
* is keyword separator
b is keyword identifiers
* is separator
c is identifier
,

(Q3) : Count words & digits:
@ output:
Enter input: Roll Number 273
Count of words: 2
Count of Numbers: 1

(Q4) Vowels and consonants:
input: good
Count of Vowels: 2
Count of Consonants: 2
4 Completed

code:
a|e|i|o|u|A|E|Z|O|U + d++ ;
[a-z A-Z] .{d++}

In printf ("Count of vowels: %d
Count of consonants: %d", c
:)

Output

```
Enter a sentence:  
I was born in 2003.  
No of characters and digits are 10 and 4  
Hello123  
No of characters and digits are 5 and 3
```

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

Code

Output window of quick take test (22)
Date _____
Page _____

inf abc;
int is keyword.
a is keyword identifier.
+ is keyword separator.
b is keyword identifier.
, is separator.
; is identifier.
+ is separator.

(Q3) : (Count words & digits.)
@ Output:
Enter input: Roll Number 243
Count of words = 2
Count of Numbers = 1

(Q4) Vowels and consonants:
input: good
Count of vowels: 2 (Completed)
Count of consonants: 2
code:
%e /i/o/u/A/E/2/0/v + d++;
(a-z A-Z) .(d++)

In dprintf ("Count of vowels: %d
Count of consonants: %d", c
:?)

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  
AC
```

Lab 2

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

Code

0912. Semantics document (1)

number is 91 dot 91

go is character 91 is digits

hi is characters

* If we remove *+ from [a-z A-Z] we get individual line of output for each character.

82) Input 'int a.b.c.'

```
%{  
#include <stdio.h>  
%P  
%y  
int float lchar lprintz {"y.c is keyword"  
"y.y is identifier"};  
[a-zA-Z]* "% " "Identifiers";  
%y "%y is identifier"  
yywrap();  
}  
int main()  
{  
    lprintz ("Enter input:");  
    yylex();  
    return 0;  
}
```

Output

I will make things happen.
No of words in the sentence are 5.

2.2 Write a program in lex to demonstrate regular definition.

Code

Done
Page

```
1) Command Terminal
Desktop $ vi programs.y

%{ #include <stdio.h> %} // Definition
S-#-+*/\n Rule action
[a-zA-Z]* printf ("%s is characters"
[0-9]* printf ("%s is digits", yytext)
int yywrap()
{
    /* do other stuff here */
}
int main()
{
    printf ("Enter input: ");
    yylex();
    return 0;
}

Open Terminal : (Enter commands)
Desktop $ lex programs.y
Desktop $ gcc lex.yy.c
Desktop $ ./a.out
```

Output : Enter Input:
good morning
good is characters morning is characters
15 is digits

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

```
To take input from input.txt and output to output.txt  
/* Options noyywrap  
 * #include <stdio.h>  
 */  
int char / float { printf(yyout, "%s →  
 keyword, yytext); }  
> /; { printf(yyout, "%s → separator, yytext); }  
= { fprintf(yyout, "%s → assignment  
 operator, yytext); }  
[0-9]* { printf(yyout, "%s → digit, yytext); }  
[a-zA-Z0-9]* { printf(yyout, "%s → identifier,  
 yytext); }  
/* */  
int wrap()  
{  
    int main()  
    {  
        yyin = fopen("input.txt", "r");  
        yyout = fopen("output.txt", "w");  
        yylex();  
        return 0;  
    }
```

Output

```
*input.txt
*input.txt
1 int sum,x=2,y=3;
2 sum=x+y;

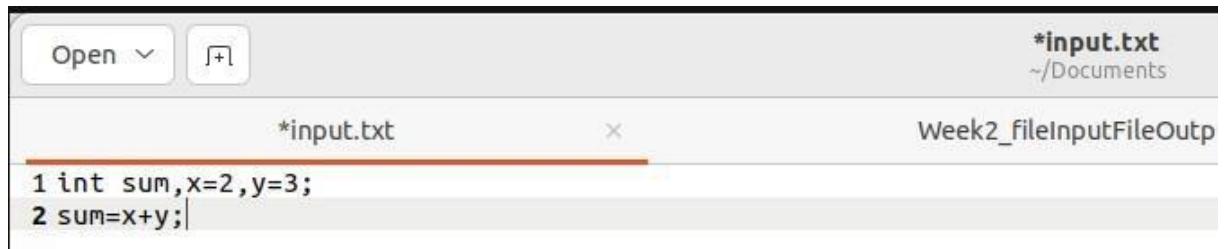
int is a keyword.
sum is an identifier.
, is a separator.
x is an identifier.
= is an assignment operator.
2 is/are digit(s).
, is a separator.
y is an identifier.
= is an assignment operator.
3 is/are digit(s).
; is a delimiter.
sum is an identifier.
= is an assignment operator.
x is an identifier.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

```
To take input from input.txt and output to output.txt  
→ /* Options noyywrap  
   #include <stdio.h>  
   yytext  
   int char / float { fprintf(yyout, "%s →  
   keyword", yytext); }  
   | ; { fprintf(yyout, "%s → separator", yytext); }  
   = { fprintf(yyout, "%s → assignment  
   operator", yytext); }  
   [0-9]* { fprintf(yyout, "%s → digit", yytext); }  
   [a-zA-Z0-9]* { fprintf(yyout, "%s → identifier,  
   yytext); }  
   ./.  
   int wrap()  
   {  
   int main()  
   {  
       yyin = fopen("input.txt", "r");  
       yyout = fopen("output.txt", "w");  
       yylex();  
       return 0;  
   }
```

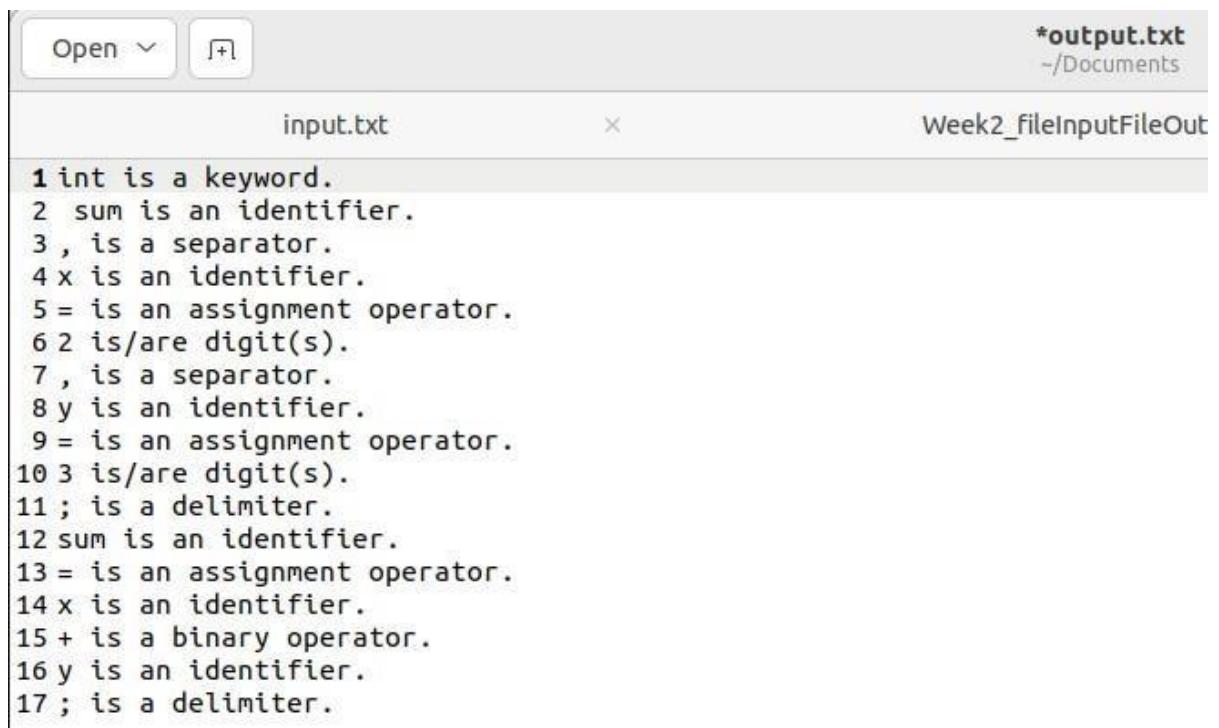
Output



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

```
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.
```

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

Code

Command Terminal
Desktop & vi programs.
/* #include <stdio.h> */ // Definition
S-+1-1x Rule action
[a-zA-Z]* printf ("%s is characters",
ytext);
[0-9]* & printf ("%s is digits", ytext);
int yywrap()
{
 /* digits should fit */
 int main()
{
 printf ("Enter input:");
 yylex();
 return 0;
 }
}
Open Terminal: (Enter commands)
Desktop & lex programs 1.
Desktop & gcc lex.y.c
Desktop & la.out
Output: Enter Input:
good morning
good is characters morning is characters
18 is digits

Output

```
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

Lab Program 1

Q) Write a program in LEX to recognize Floating Point Numbers.

```
#include <stdio.h>
int yywrap()
{
    int main()
    {
        printf("Enter the number ");
        yylex();
        return 0;
    }
}
```

off

Enter number
23.6
23.6 is a floating-point number.

45

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

slu123

(1) #include <stdio.h>
int flag = 0;
. /.
and/or/but/because/if/then/nevertheless
{
 int gwrap();
 if (flag == 0)
 printf ("Simple sentence\n");
 else
 printf ("Compound sentence\n");
 return 1;
}

int main ()
{
 printf ("Enter the sentence : \n");
 gwrap();
}

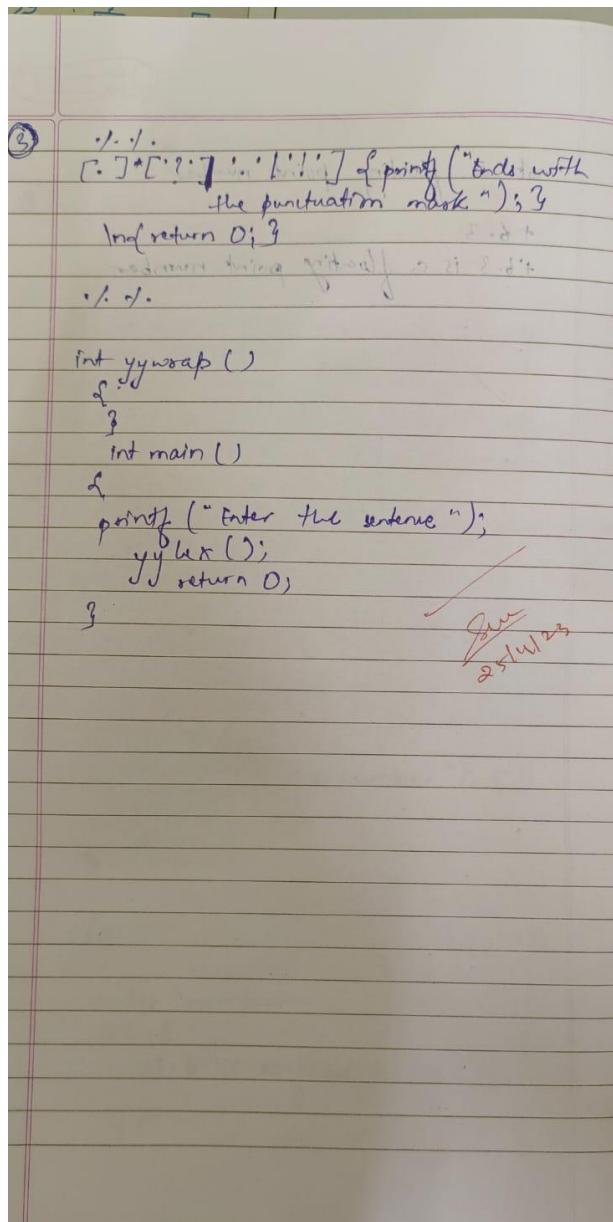
Output
hello
Simple sentence
Hello and bye → compound sentence.

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

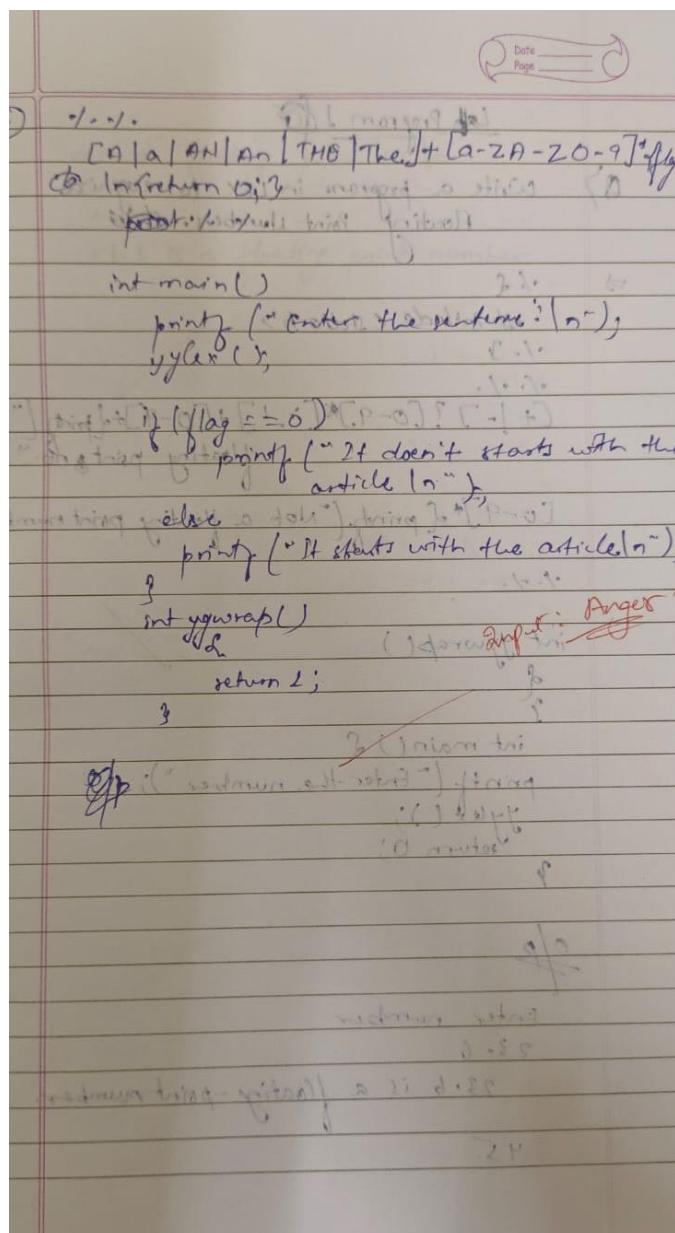


Output

```
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:  
An apple a day keeps the doctor away.  
Starts with an article!  
  
Enter a sentence:  
A book is lying on the table.  
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

Q6) Print valid string for alphabetical string as invalid for alpha-numeric string.

```
#include <stdio.h>
int yywrap();
int yytext;
int yylex();

main()
{
    int i;
    char c;
    for(i=0;c=getchar();i++)
        if(c=='a' || c=='e' || c=='i' || c=='o' || c=='u')
            printf("%c is vowel\n",c);
        else
            printf("%c is consonant\n",c);
}
```

Q7) Program to take input from a file and print output.

```
#include <stdio.h>
int yywrap();
int yytext;
int yylex();

main()
{
    int i;
    char c;
    for(i=0;c=getchar();i++)
        if(c=='a' || c=='e' || c=='i' || c=='o' || c=='u')
            printf("%c is vowel\n",c);
        else
            printf("%c is consonant\n",c);
}
```

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
^C
```

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

Code

```
#include <stdio.h>
#include <yyFlex.h>
int yywrap() { return 0; }

int main()
{
    printf("Enter the sentence: ");
    yytext = yylex();
    if (yytext[0] == '+' || yytext[0] == '-')
        printf("%s is signed\n", yytext);
    else
        printf("%s is unsigned\n", yytext);
}
```

Output

```
Enter a number:
123
Unsigned number!

-123
Signed number!

+123
Signed number!

^C
```

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 12/12/23
Page

LAB - 4

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

```

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

char str[200];
int ytext;
int yytext(char *s);

yytext(char *s)
{
    if (*s == ' ') {
        *s = '\n';
        s++;
    }
    while (*s != '\0') {
        *s = ' ';
        s++;
    }
}

yytext()
{
    if (*str == '\0') {
        *str = ' ';
        str++;
    }
    while (*str != '\0') {
        *str = ' ';
        str++;
    }
    return 0;
}

int main()
{
    char filename[200];
    printf("Enter name of file: ");
    scanf("%s", filename);
    yyin = fopen(filename, "r");
    printf("Enter name of file of write: ");
    scanf("%s", filename);
    yyout = fopen(filename, "w");
    yylex();
}

```

Output

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

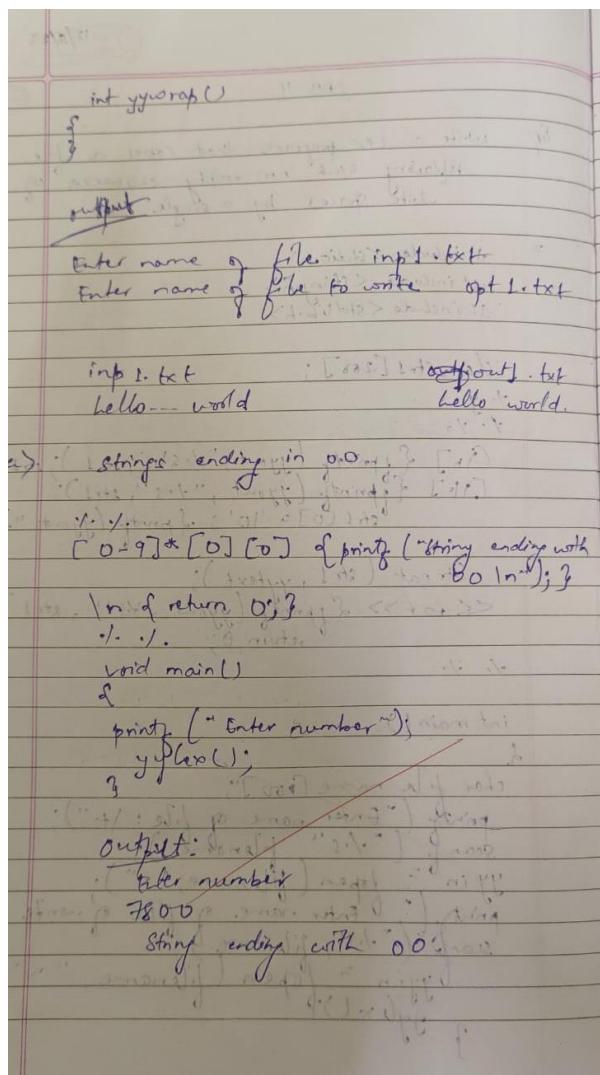
Printed!

```
Open ▾  ↗
print.txt
~/Documents
1 Hello World
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



int yywrap()
{
 /* code */
}

output:
Enter name of file to input: input.txt
Enter name of file to write: opt1.txt

input.txt : [Hello] world output.txt
Hello world.

a) /* strings ending in 00 */
/* (0-9)*00 */
[0-9]*[0][0] printf ("String ending with
00\n");
return 0;
/* */

void main()
{
 printf ("Enter number");
 yylex();
}

output: Enter number
7800
String ending with 00.

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 2's.

Code

26) String with three consecutive 2's.

→ $[0-9]^* [2] [2] [2] [0-9]^*$ if print("String contains 222")
if string contains 222 then
return 0;
else return 1;

Output

Enter Number: 3622295
String contains 222

27) Set of all numbers ("digit" whose sum is 9)

→ digits [0-9]
→ $3 \text{ digit } + 2 \text{ digits } + 2 \text{ digits } + 3$
 $\text{for } i = 0 \text{ to } \text{length} - 1; j >= 0; i++$
 $\text{value} += (\text{ytext}[i] - 48);$
~~if value == 9~~
~~flag = 1;~~
[In] return 0;
else

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

output: 1341
success

2d) set of all strings beginning with 101,
binary representation of an integer
or congruent to zero modulo 5.

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    int value = 0;
    int flag = 0;
    int i = 0;
    int j = 0;
    int n = 0;
    cout << "Enter a string: ";
    string yytext;
    cin >> yytext;
    n = yytext.length();
    for (i = 0; i < n; i++) {
        if (yytext[i] == '0') {
            value = value + pow(2, i);
        } else if (yytext[i] == '1') {
            value = value + pow(2, i);
        } else if (yytext[i] == '2') {
            value = value + pow(2, i);
        } else if (yytext[i] == '3') {
            value = value + pow(2, i);
        } else if (yytext[i] == '4') {
            value = value + pow(2, i);
        } else if (yytext[i] == '5') {
            value = value + pow(2, i);
            flag++;
        } else if (yytext[i] == '6') {
            value = value + pow(2, i);
        } else if (yytext[i] == '7') {
            value = value + pow(2, i);
        } else if (yytext[i] == '8') {
            value = value + pow(2, i);
        } else if (yytext[i] == '9') {
            value = value + pow(2, i);
        }
    }
    if (value % 5 == 0) {
        cout << "Valid string." << endl;
    } else {
        cout << "Not a valid string!" << endl;
    }
}
```

Output:
101
success

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

output: 1010
13 41
success

2d) set of all strings beginning with 1st, binary representation of an integer is congruent to zero modulo 5.

```
if (y[0] == '1') {  
    for (i = y.length - 1; i >= 0; i--) {  
        value = value + (y[i] - '0') *  
            pow(2, i);  
        j++;  
    }  
    if (value % 5 == 0)  
        cout << "Congruent to zero modulo 5.";  
    else  
        cout << "Not congruent to zero modulo 5.";  
}  
else  
    cout << "Not a binary number.";
```

10 1 (Programmer) 10
success

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

Output ~~and various sample codes~~

1341
success

2d) set of all strings beginning with 1~~0~~,
binary representation of an integer
or congruent to zero modulo 5.

→ $\{1, 0\}^*$ { for ($i \in \text{yyLength}-1$; $i >= 0$; $i--$)
 values = value + (yyText[i] - 48)*
 pow(2, i);
 j++;
 if (value % 5 == 0)
 if (j == 1);
 else [In] return output + "1" }
~~Output~~
101 (P-1341)
success

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

26) String with three consecutive 2's

```
→ void stringWithThreeConsecutiveTwos(string s) {
    if (s.length() < 3) return;
    for (int i = 0; i < s.length() - 2; i++) {
        if (s[i] == '2' & s[i + 1] == '2' & s[i + 2] == '2') {
            cout << s;
        }
    }
}
```

Output

```
Enter 'Number': 3622295
3622295
String contains 222
```

27) Set of all numbers ("digit whose sum is 9")

```
→ digits [0-9]
→ for (int i = 0; i < 10; i++)
    for (int j = 0; j < 10; j++)
        for (int k = 0; k < 10; k++)
            for (int l = 0; l < 10; l++) {
                int value = i + j + k + l;
                if (value == 9) {
                    cout << i << j << k << l;
                }
            }
}
[In] return 0;
→
```

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.
```

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

Code

2e) 10th digit from right is 1
→ digits [0-9]
{ digits } + ! { digits }...{ digits }
 printf("%1.8s 10th digit from
 right end is 1; %s\n");
 (digits) printf ("condition not satisfied ");

Output
1 2 3 4 5 6 7 8 9 1
10th symbol from right end is 1.
2f) set of all four digital numbers, whose
individual digits are in ascending order
from left to right
→ digits [0-9]
{ digits } & { digits } & { digits } & { digits }
 for (i=0; i<yy[yy-1]; i++)
 if (yytext[i] > yytext[i+1])
 {
 if (i==0) S(8)
 return 0;
 }
Output
1 2 3 4
success!

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 _____
Page _____

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

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

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

```

Page 1/1

for (i=0; i<= strlen(input); i++) {
    if (isOperator(input[i]) || isPunctuation(input[i]))
        || isspace(input[i]) || input[i] == '\0')
    {
        buffer[j] = '0';
        if (j > 0)
        {
            if (isKeyword(buffer))
                printf("Keyword: %s\n", buffer);
            else if (isNumber(buffer))
                printf("Number: %f\n", buffer);
            else
                printf("Identifier: %s\n", buffer);
        }
        if (isOperator(input[i]) || isPunctuation(input[i]))
            printf("Operator/Punctuation: %c\n", input[i]);
        j = 0;
    } else
        buffer[j++] = input[i];
}

int main()
{
}

```

```

int isPunctuation (char ch)
{
    char punctuations[] = ".,;(){}";
    int i;
    for(i=0; i< strlen(punctuations); i++)
        if(ch == punctuations[i])
            return 1;
    return 0;
}

int isNumber (char *str)
{
    int i;
    for(i=0; i< strlen(str); i++)
        if(!isdigit(str[i]))
            return 0;
    return 1;
}

void analyze(char *input)
{
    char buffer[20];
    int i, j = 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

Date 26/12/23
Page

Lab 6

Write a program to perform Recursive Descent Parsing on the following grammar.

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

```
#include <stdio.h>
int index = 0;

void parse_A(char input_str[]);
void parse_S(char input_str[]);

void parse_A(char input_str[])
{
    if (input_str[index] == 'a')
        index++;
    else if (input_str[index] == 'b')
        printf("%c", input_str[index++]);
    else
        printf("Input string is not accepted");
}

void parse_S(char input_str[])
{
    parse_A(input_str);
    if (input_str[index] == 'c')
        parse_D(input_str);
    else
        printf("Input string is not accepted");
}
```

```

void parse_S(char input_str[])
{
    if (input_str[index] == 'c')
    {
        index += 1;
        parse_A(input_str);
        if (input_str[index] == 'd')
        {
            index += 1;
            printf("Input string is accepted.\n");
        }
        else
        {
            printf("Input string is not accepted.\n");
        }
    }
    else
    {
        printf("Input string is not accepted.\n");
    }
}

int main()
{
    char input_string[] = "cabd";
    index = 0;
    parse_S(input_string);

    return 0;
}

```

Output:

```

String : cabd
Input string is accepted.

String: cad
Input string is not accepted.

```

Output

```

1.S->cAd
2.A->ab/a
Enter any string:cabd
String is accepted by the grammar
The productions used are
S -> cAd
A -> ab

...Program finished with exit code 1
Press ENTER to exit console. []

```

Lab 7

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

Code

Pg 2:
 Modify the program so as to include operators as '+', '-', '*' as per the associativity + precedence.

Ques 1:
 #include "y.tab.h"
 extern int yyleval;
 . . .

$[0-9]^+ \cdot f$ yyleval = atoi(yytext);
 return digit;

}
 [- +] ; i all what? } string
 [n] return 0; }
 return yytext[0]; }
 . . .

int yywrap()
 { : ("quit") string

Ques 2:
 #include <ctype.h>
 #include <stdio.h>
 #include <stdlib.h>
 . . .

1. token digit
 1. left '+' '-'
 1. right '*' '/'
 1. right '^'

```

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

    E : E { printf ("+"); }
    | E - E { printf ("-"); }
    | E * E { printf ("*"); }
    | E / E { printf ("/"); }
    | E % E { printf ("%"); }
    | ( E ) { digit & printf ("%.d", &E); }

    ; of printing 3 parts = Double [0-a]
    int main()
    {
        printf ("Enter the infix expression");
        yyparse();
        yyerror();
        printf ("Error");
    }

    Output:
    Enter Infix Expression
    4 + 3 * (8 - 2) / 2 ^ 2
    4 + 3 8 2 - 1 * 2 2 ^ /
    (4.0000) result 16
    16

```

Output

```

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

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

```

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

Code

Page

LAB-6

Q7 Write a YACC program to recognize string
 $\{a^n b | n \geq 5\}$

→ P1.y

```
#include "y.tab.h"
extern int yyval;
yyval = yytext[0];
if (yyval == 'A') {return A;}
else if (yyval == 'B') {return B;}
else if (yyval == '\n') {return NL;}
else {return yytext[0];}
```

int yywrap () {return 1;}

P1.y

```
#include <stdio.h>
#include <stdlib.h>
K: tokens A B NL
stmt: A A A A S B N L
printf ("valid string\n"); exit(0)
```

Output

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



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

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

Code

Lab 7 hands on
Arithmetic expression parser

Pl. L

```
#include "y.tab.h"
extern int yyval;
int yylex();
int yyparse()
{
```

Pl. y

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



```

yyposse();
    return 0;
}

int yyerror()
{
    printf("NFTW: Error(%d)\n", yyline);
    int mknodel(int le, int re, char val[10])
{
    strcpy(symtree[le].val, val);
    symtree[le].lc = le;
    symtree[le].rc = re;
    symtree[le].node += 1;
    return symtree[le].node - 1;
}

void yyprinttree(int currnode)
{
    if (currnode == -1)
        return;
    if (symtree[currnode].lc == -1 & symtree[currnode].rc == -1)
        printf("Digit Node -> index : %d,\n"
               "Value : %c, Left Child Index : \n"
               ". . . Right Child Index : %d\n",
               currnode, symtree[currnode].val,
               symtree[currnode].lc,
               symtree[currnode].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 16/11/24
Page _____

LAB-8

Q7 Proj 1 :- (Use YACC to convert :: infix expression to postfix expression)

```
#include "y.tab.h" // header file
/* of #include "y.tab.h" */
extern int yyval;
/* 3 */
/* 0 */
E : E '+' T {printf ("+"); }
| digit
| '-' digit
| '*' digit
| '/' digit
| '^' digit
| '(' digit ')'
| digit
| '.' digit
| int yywrap ()
```

Py.y corresponds with .c file

```
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
/* taken digit */
/* */
S : of printf ("In In");
| E : E '+' T {printf ("+"); }
```

Program

```

T : T * T { printf ("%*"); }
| F d u c t o f i n f i x e x p r e s s i o n
|
F : ' ( ' E ')
| digit { printf ("%d", #1); }
| . . .
int main()
{
    printf ("Enter infix expression");
    yyparse();
}
yyerror(): // creates yyerr
{ printf ("Error");
}

```

Output:

Enter the infix expression:
~~2+6*3+4~~ To do
~~263.*+4+~~ And with
~~< backslash > showing the~~

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 23/11/24
Page

Lab - 9 (c) write 3 address code

Use YACC to generate 3-Address code for a given expression

```
#include "y.tab.h"
int yywrap();
char yytext[20];
int yylex();
int var_count = 0;
```

#include <stdio.h>
#include <stdlib.h>
#include <y.tab.h>
extern int yyval;
extern char yytext[20];

int yylex()
{
 if (yytext[0] == '0' || yytext[0] == '1' || yytext[0] == '2' || yytext[0] == '3' || yytext[0] == '4' || yytext[0] == '5' || yytext[0] == '6' || yytext[0] == '7' || yytext[0] == '8' || yytext[0] == '9')
 return digit;
 else if (yytext[0] == '+' || yytext[0] == '-' || yytext[0] == '*' || yytext[0] == '/')
 return op;
 else if (yytext[0] == '=')
 return assign;
 else if (yytext[0] == 'x' || yytext[0] == 'y' || yytext[0] == 'z')
 return id;
 else if (yytext[0] == ' ')
 return space;
 else
 return error;
}

int yywrap()
{
 return 0;
}

#include <math.h>
#include <ctype.h>
#include <stdio.h>
int var_count = 0;

~~char iden [20]; (n)~~

1. $\{ \}$

shift taken id \rightarrow var_id \rightarrow \$3\$
 2. taken digit
 3. $=$.

S: id \equiv E; printf ("%s = %d\n", idem, var_id);
 $\quad \quad \quad +$ E -> S

E: E + T $\quad \& \quad$ T $\{ \$ \} =$ var_int; var_int++; printf ("%d = %d + %d; %d\n", \$1, \$2, \$3, \$4);
 $\quad \quad \quad +$ E -> E

| E \neg T $\{ \$ \} =$ var_int; var_int++; printf ("%d = %d - %d; %d\n", \$1, \$2, \$3, \$4);
 $\quad \quad \quad -$ E -> E

| T $\{ \$ \} =$ \$1; ?

| ;
 T: T * F $\{ \$ \} =$ var_int; var_int++; printf ("%d * %d = %d\n", \$1, \$2, \$3); ?
 T: T / F $\{ \$ \} =$ var_int; var_int++; printf ("%d / %d = %d\n", \$1, \$2, \$3); ?

F: P $\{ \$ \} =$ var_int; var_int++; printf ("%d + %d = %d\n", \$1, \$2, \$3); ?

| P $\{ \$ \} =$ \$1; ?

| P: ('E') $\{ \$ \} =$ \$2; ?
~~| digit & \$1 = var_int; var_int++; printf ("%d = %d\n", \$1, \$2); ?~~

| ;
 int main ()

```

2
var_cnt = 0;
printf ("Enter an expression : [n]");
yyparse();
return 0;
}
yyerror()
{
    printf ("error");
}

```

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
```