LAB FILE

of

Compiler Design Laboratory

(CSE606)

Bachelor of Technology (CSE)

By

Dhairya Gohil (22000807)

Third Year, Semester 6

Course In-charge: Prof. Vaibhavi Patel

Department of Computer Science and Engineering

School Engineering and Technology

Navrachana University, Vadodara

Spring Semester

(2025-2026)

1. Implement the automata to

a. Recognize strings starts with ‘a’ over {a, b}.

b. Recognize strings end with ‘a’.

c. Recognize strings end with ‘ab’. Take the input from text file.

d. Recognize strings contains ‘ab’. Take the input from text file.

```python
with open('automata.txt','r') as file:
    string = file.read()

while True:
    print('1. Starts with a\n2. Ends with a\n3. Ends with ab\n4. Contains ab')
    a = input('Enter the string(1-4) : ')
    if a=='1':
        if string.startswith('a'):
            print('{} starts with \'a\'.\n'.format(string))
        else:
            ('{} does not starts with \'a\'.\n'.format(string))
    elif a=='2':
         if string.endswith('a'):
             print('{} ends with \'a\'.\n'.format(string))
         else:
             print('{} does not ends with \'a\'.\n'.format(string))
    elif a=='3':
        if string.endswith('ab'):
            print('{} ends with \'ab\'.\n'.format(string))
        else:
            ('{} does not ends with \'ab\'.\n'.format(string))
    elif a=='4':
        if 'ab' in string:
            print('{} contains \'ab\'.\n'.format(string))
            count = 0
            for i in range(len(string)-1):
                if string[i]=='a' and string[i+1]=='b':
                    count += 1
            print('Total count :',count)
        else:
            print('{} does not contains \'ab\'.\n'.format(string))

    elif a=='quit':
        break
```

2.a. Write a program to recognize the valid identifiers.

```
with open('dhairya.txt','r') as file:
    string = file.read()

string = string.replace('(',' ').replace(')',' ').replace(':',' ')
word = set(string.split())

keywords = ['for','if','else','in']

for i in word:
    if i.isidentifier() and i not in keywords:
        print(i)
```

2.b. Write a program to recognize the valid operators.

```
with open('dhairya.txt','r') as file:
    string = file.read()

string = set(string.split())

count = 0
arithmatic = []
assignment = []
relation = []
```

```
logical = []

bitwise = []

unary = []

for i in string:

    if i in ['+','-','*','/']:

        count += 1

        arithmatic.append(i)

        print(i)

    elif i in ['<','>','<=','>=','==','!=']:

        count += 1

        relation.append(i)

    elif i in ['&&','||','!']:

        count += 1

        logical.append(i)

    elif i in ['++','--']:

        count += 1

        unary.append(i)

    elif i in ['=','+=','-=','*=','/=','%=']:

        count += 1

        assignment.append(i)

    elif i in ['&','|','~','<<','>>']:

        count += 1

        bitwise.append(i)

    else:

        pass


print('Operators found :',count)

print('Arithmatic :',arithmatic)
print('Relational :',relation)

print('Assignment :',assignment)

print('Logical :',logical)

print('Bitwise :',bitwise)

print('Unary :',unary)
```

Other program:

```
with open('dhairya.txt','r') as file:

    string = file.read()


string = string.replace('(',' ').replace(')',' ').replace(':',' ')

word = set(string.split())


for i in word:

    if i in ['+','-','*','/','%']:

       print(i,'is Arithmatic operator.')

    elif i in ['>','<','<=','>=','==','!=']:

        print(i,'is Relational operator.')

    elif i in ['&&','||','!']:

        print(i,'is Logical operator.')

    elif i in ['=','+=','-=','*=','/=','%=']:

        print(i,'is Assignment operator')

    elif i in []:

        print(i,'is Unary operator')
```

2.c. Write a program to recognize the valid number.

```python
with open('dhairya.txt','r') as file:
    string = file.read()

for ch in ['=','+','-','*','/',':','(',')','\n','<','>']:
    string = string.replace(ch,' ')

word = string.split()
print(word)

count = 0
for i in word:
    try:
        num = float(i)
        count += 1
    except ValueError:
        pass

print('Total numbers found :',count)
```

Other program:

```python
with open('dhairya.txt','r') as file:
    string = file.read()

string = string.replace('(',' ').replace(')',' ').replace(':',' ')
word = set(string.split())
```

```
for i in word:
    try:
        float(i)
        print(i,'is a number')
    except ValueError:
        pass
```

2.d. Write a program to recognize the valid comments.

```
with open('dhairya.txt','r') as file:
    string = file.read()
n = len(string)
i = 0
count = 0
comment = []

while i<n:
    #multi line comment : ''' to '''
    if string[i:i+3] == "'''":
        end = string.find("'''",i+3)
        if end != -1:
            comment.append(string[i+3:end])
            count += 1
            i = end+3
        else:
            break
    #multi line comment : """ to """
    elif string[i:i+3] == '"""':
        end = string.find('"""',i+3)
        if end != -1:
            comment.append(string[i+3:end])
            count += 1
            i = end + 3
        else:
            break

    #single line comment
    elif string[i] == '#':
        end = string.find('\n',i)
        if end == -1:
            end = n
        comment.append(string[i+1:end])
        i += 1
        count += 1


    else:
        i += 1


print(count)
print(comment)
```

4.a. Write a Lex program to take input from text file and count no of characters, no. of lines & no. of words.

```
%{

#include <stdio.h>

int c=0,w=0,l=0;

%}

%%

[^\n\t]+ {w++; ; c+=yyleng;}

\n {l++; c++;}

.  {c++;}

%%


int main() {

yyin = fopen("dhairya.txt","r");

yylex();

printf("\nCharacters : %d",c);

printf("\nWords : %d",w);

printf("\nLines : %d",l);

}

int yywrap(){return(-1);}
```

4.b. Write a Lex program to count number of vowels and consonants from a given input string.

```
%{

#include <stdio.h>

int v=0, c=0;

%}

%%

[aeiouAEIOU] {v++;}

[bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ]  {c++;}

[^a-zA-Z] ;

%%


int main() {

yyin = fopen("dhairya.txt","r");

yylex();

printf("\nVovels : %d",v);

printf("\nConsonants : %d",c);

return 0;

}

int yywrap(){return (-1);}
```

4.c. Write a Lex program to print out all numbers from the given file.

```
%{
#include <stdio.h>
int count = 0;
%}
%%
[0-9]+(\.[0-9]+)?([eE][+-]?[0-9]+)? {printf("\n%s This is valid number",yytext);
count++;}
count++
\n ;
. ;
%%

int main() {
yyin = fopen("input.txt","r");
yylex();
printf("\n\ncount is : %d",count);
return 0;
}
int yywrap(){return(1);}
```

4.d. Write a Lex program which adds line numbers to the given file and display the same onto the standard output.

```
%{
#include <stdio.h>
int line = 1;
%}

%%
.+ {fprintf(yyout,"%d -> %s",line,yytext); line++;}
%%

int main() {
yyin = fopen("dhairya.txt","r");
yyout = fopen("output.txt","w");
yylex();
printf("Done...");
return 0;
}
int yywrap(){return(-1);}
```

4.e. Write a Lex program to printout all HTML tags in file.

```
%{

#include <stdio.h>

int num = 0;

%}

%%

"<"[A-Za-z0-9]+">" {printf("\n%s is valid html tag.",yytext); num++;}

"<!--"(.|\n)*"-->" {   }

\n ;

. ;

%%


int main() {

yyin = fopen("dhairya.txt","r");

yylex();

printf("\nTotal tages : %d",num);

return 0;

}

int yywrap(){return(-1);}
```

5.a. Write a Lex program to count the number of comment lines from a given C program. Also eliminate them and copy that program into separate file.

```
%{

#include <stdio.h>

int single=0, multiple=0;

%}

%%

"//".* {single++;}

"/*"([^*]|\n)*"*/"  {multiple++;}

%%


int main() {

yyin = fopen("cde.txt","r");

yylex();

printf("\nsingle line comment %d",single);

printf("\nMultiple line comment %d",multiple);

return 0;

}

int yywrap(){return(-1);}
```

5.b. Write a Lex program to print keywords, identifiers, operators, numbers from a given C program.

```
%{

#include <stdio.h>

int num=0,op=0,id=0,key=0;

%}

%%

"//".* ;

"/*"([^*|\n])*"*/" ;

[0-9]+(\.[0-9]+)?([eE][+-]?[0-9]+)? {num++;}

"+"|"-"|"*"|"=" {op++;}

"a"|"b"|"printf" {id++;}

"int"|"main"|"return" {key++;}

%%


int main() {

yyin = fopen("cde.txt","r");

yylex();

printf("\nIdentifiers : %d",id);

printf("\nNumbers : %d",num);

printf("\nKeywords : %d",key);

printf("\nOperators : %d",op);

return 0;

}

int yywrap(){return(-1);}
```

7. Program to implement Recursive Descent Parsing in C.

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

char inp[100];
int d=0;

void match(char t) {
    if (inp[d]==t) {
        d++;
    }
        else {
            printf("Error");
            exit(0);
        }
    }

void E();
void E_prime();

void E() {
    if (inp[d]=='i') {
        match('i');
        E_prime();
    }
}
```

```
void E_prime() {

    if (inp[d]=='+') {

        match('+');

        match('i');

        E_prime();

    }

    else if (inp[d]=='-') {

        match('-');

        match('i');

        E_prime();

    }

    else {

        return;

    }

}


int main() {

    printf("Enter the string : ");

    scanf("%s",inp);


    E();


    if (inp[d]=='$') {

        printf("Success");

    }

    else {

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

8.

a. To Study about Yet Another Compiler-Compiler

b. Create Yacc and Lex specification files to recognizes arithmetic expressions involving +, -, * and / .

c. Create Yacc and Lex specification files are used to generate a calculator which accepts integer and float type arguments.

(a)

LEX FILE

```
%{
#include "pfix.tab.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
%}
```

```
%%
[0-9]+ {yylval.num = atoi(yytext); return INTEGER; }
[a-zA-Z_][a-zA-Z0-9_]* {yylval.str = strdup(yytext); return ID; }
[+\-*/()] {return yytext[0];}
[\n] {return '\n';}
[ \t\r] ;
.  {printf("Error! Invalid character :'%s'\n",yytext);}
%%

int yywrap() { return 1;}
```

YACC FILE

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int yylex(void);
void yyerror(char *);
%}

%union {
        char *str;
        int num;
}

%token <num> INTEGER
```

```
%token <str> ID

%type <str> F

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

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

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

F: INTEGER       { printf("%d ", $1); }
 | ID            { printf("%s ", $1); }
%%

void yyerror(char *s){
    fprintf(stderr, "Error: %s\n", s);
}

int main() {
    yyparse();

    return 0;

}
```

(b)

LEX FILE

```
%{

#include "arithmatic.tab.h"

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void yyerror(char *);

%}


%%

[0-9]+ return num;

[-+*\n] return *yytext;

[ \t] ;

. {printf("Invalid character\n");}

%%


int yywrap() {return 1;}
```

YACC FILE

```
%{
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void yyerror(char *);
int yylex(void);
%}

%token num

%%
S: E '\n'    {printf("Valid syntax"); return 0;}

E: E '+' T   { }
 | E '-' T   { }
 | T         { }

T: T '*' F   { }
 | F         { }

F: num       { }
%%
```

```
void yyerror(char *s) {
        printf("%s\n",s);
}

int main() {
        yyparse();
        return 0;
}
```

(c)

LEX FILE

```
%{
#include <stdio.h>
#include <stdlib.h>
int yylex(void);
void yyerror(char *);
#include "calc.tab.h"
%}

%%
[0-9]+ {yylval = atoi(yytext); return num;}
[-+*\n] {return *yytext;}
[/()] {return *yytext;}
[ \t] { }
. {printf("Invalid character");}
```

```
%%

int yywrap() {return -1;}
```

YACC FILE

```
%{
#include <Stdio.h>
void yyerror(char *);
int yylex(void);
%}

%token num

%%
S: E '\n'   {printf("%d\n",$1); 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: '('E')'  {$$ = $2;}
```

```
 | num      {$$ = $1;}
%%


void yyerror(char *s) {
        fprintf(stderr, "%s\n",s);
}


int main() {
        return yyparse();
}
```