EXPERIMENT : 1

Aim : Study the Process translation & compiler.

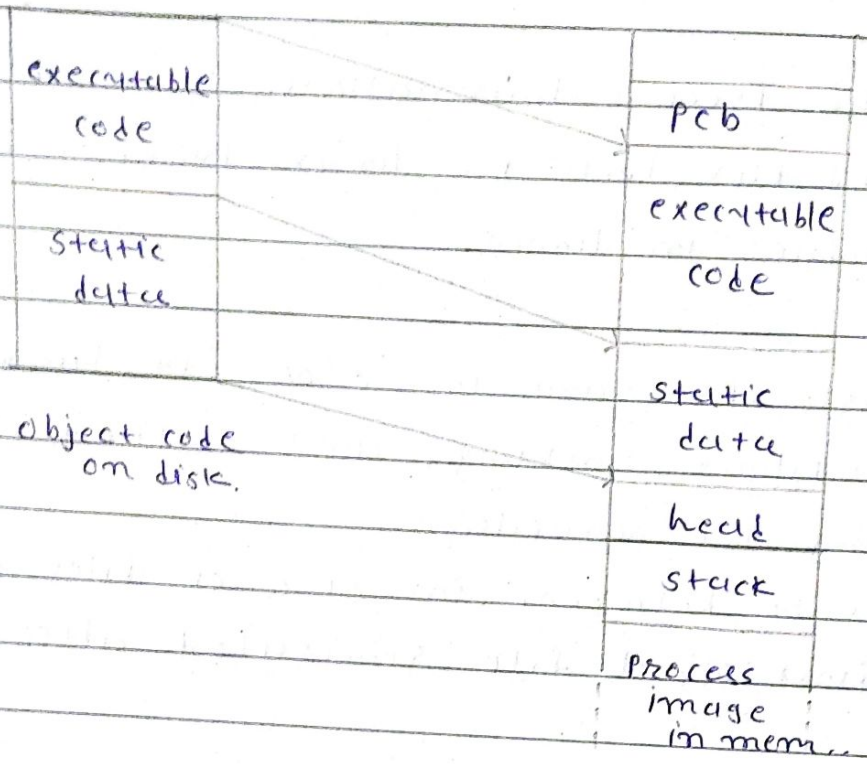
Tools : yacc and flex tools in linux, turbo-c or gcc compiler in linux.

(1) write a single program in c or c++ language & observe compiling, linking, and testing loading process using gcc compiler.
(find differance between in size of files in generated.) (find which file generated after each step).

⇒ loading :
Befor a program can execute as a process, the program load module must be loaded into main memory. in this step all the comments are removed and headers are expanded.

program :

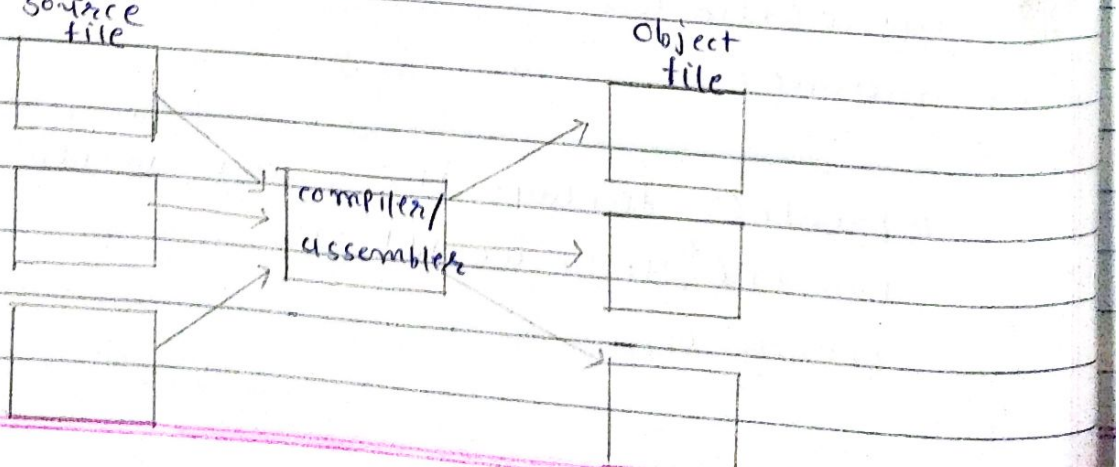
```
# include <stdio.h>
int main()
  {
    // printf() displays the string inside quotation
    printf("Hello world!");
    return 0;
  }
```



Run program:

=> CPP hello.c
=> I "hello.c"
=> I "<built-in>"
=> I "<command-line>"
=> I "hello.c"
⋮ ⋮

=> Compiling :-



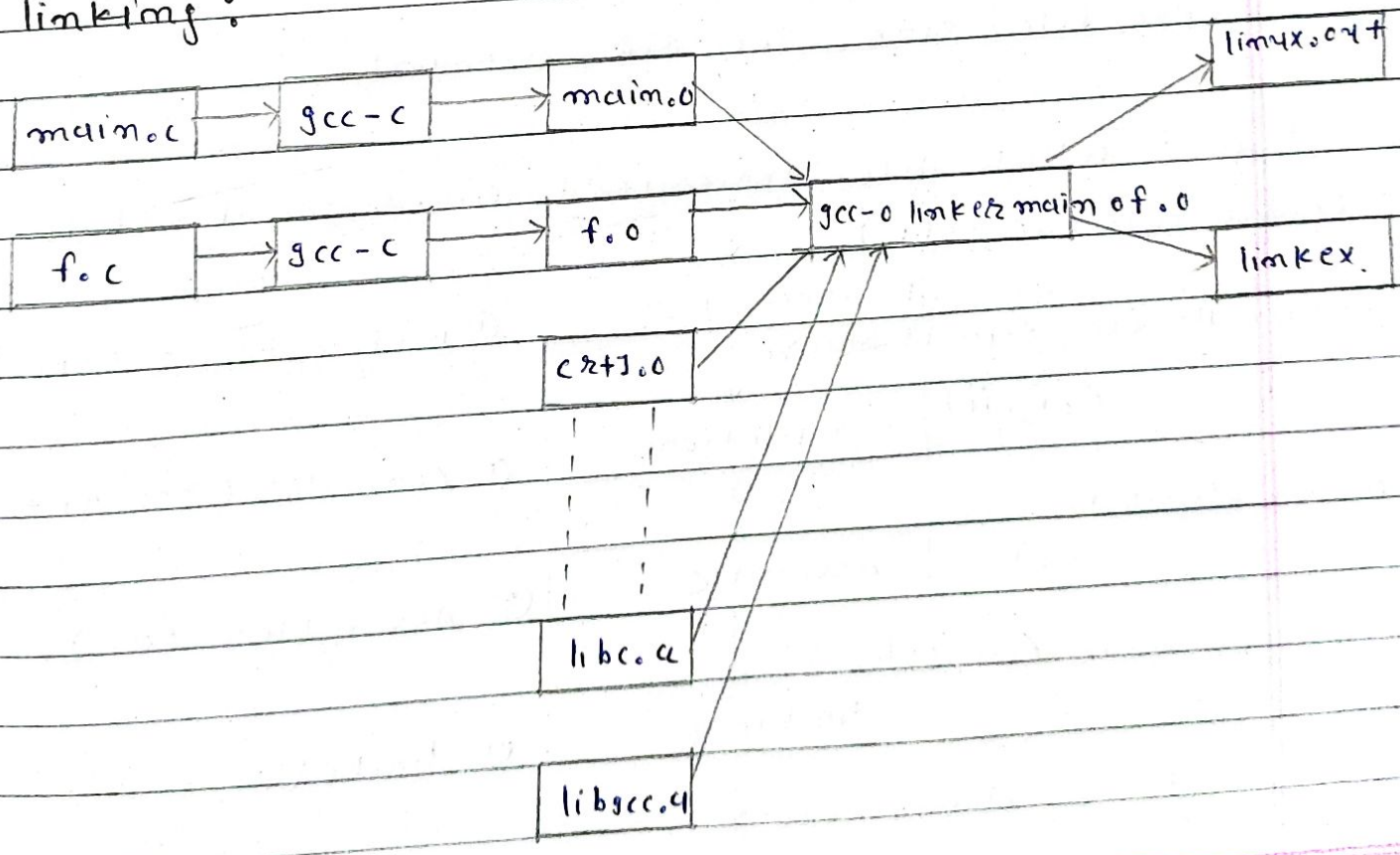
> gcc hello.c
// Compile & linux source file hello.c into executable a.exe (windows)

To run the program:
> a

run:
> gcc hello.c
> a

Hello world!

=> linking:



> gcc -o hello.exe hello.c

// compile and linux Source file hello.c into executable hello.exe

> hello

output

Hello word!

+ find differance in size of files in generated

cpp file size depend on program

.obj file size 400 MB around

.exe file size 300 MB around

+ find which file generated after each step.

Source code (.c, .cpp)

↓

Preprocessing — ① preprocessor (cpp)

include Headers, Expand Macro

(.i, .ii)

↓

compilation — ② compiler (gcc, g++)

Assembly code

(.s)

↓

Assemble — ③ Assembler (as)

Machine code (.o, .obj)

↓

linking — ④ linker

↓

executable machine code

(.exe)

(2) Now study the question & give the answers.

2.(1) define the following (1) compiler (2) interpretter (3) linker (4) lodder.

⟹ (1) compiler: Compilers are computer software that translates high-level language program into assembly language that can be understood by a computer's CPU. This is one language processor that reads the complete source program in one go and translates it into an equivalent program in machine language.

Interpreters: An interpretter directly executes instructions written in a programming language without previously converting them to an object code or machine code. Interpreted code will show an error as soon as it hits a problem, so it's easier to debug than compiled code.

linkers: linkers use tool that message the obje
Produced by separate compilation or ass
and creates an executable file. Linker is
program in a system which help to
object modules of program into single ob
file. It performs the process of linking

loaders: It is part of the os that brings an exe
file residing on disk into memory and
starts it running

→ Steps followed by loaders:

- Read exe file's header
- Create a new address space for the program
- Copies instructions and data into address spac
- Copies arguments passed to the program on
stack
- Initializes the machine registers including
- Jumps to a startup routine & calls the progs
main routine.

2.2 gcc

① what is gcc explain it's feature.

↳ The GNU Compiler collection, commonly known as GCC, is set of compilers and development tools available for Linux, windows, various BSDs, and a wide assortment of other OS.

↳ It includes support primally for C and C++ and includes objective C, GO, fortran and D.

↳ The free Software foundation (FSF) distributed GCC as the free software under the GNU General public License.

* features :

↳ It is a portal-table compiler - runs on most of all platform.

↳ It supports processors used in personal computers as well as microcentrollers, DSPs and 64-bit CPUs.

↳ GCC has mutiple language fucntends, for parsing different languages.

↳ GCC is free Software this means you have the freedom to use and to modify GCC, as with all GNU software.

↳ GCC has modulers design, allowing support for new languages and architectures to be added.

(2) In which language its gcc written?

↳ The GNU compiler collection (GCC) from its inception, written in C and compiled as C compiler. Later, effort made to com it by C++ compiler so that it can tak the advantages of C++

(3) What standard optimization are supported by gcc.

↳ The overall compiler optimization level is con by the command line option -On where n the required optimization level as follows.

↳ -O0 (default) : No optimization is performed gives the cleareste view for level debugging.

↳ -O1 : It not requires size versus speed including function inlining It can actually produce faster compile the because resulting files are Smaller

↳ -O2 : This enable additional optimizations as instruction Scheduling. Also speed size implication is not used.

- O3 : It enables aggressive function inlining and can therefore increase the speed at the expense of image size

- Os : It attemets to minimize the size of the image even at the expense of sped

- -funroll - loops : This option is independent of the -On option and enables loop unrolling It increase codesize.

2.4 explain following option of gcc size<exefilename> objdump -hzt<objfile>, -o, -s, -c, -savetemps, -d, -v, -c, -static

(i) gcc -o/-O option flags

o. gcc-o option flag (small o)

writes the build output to an Output file

Syntax:

$ gcc [option][sourcefile][objectfiles] -o output

Example : myfile.c

```
#include<stdio.h>
void main(){ printf("program run\n");
}
```

$ gcc myfile.c -o myfile
$ ./myfile
program run

b. gcc -O option flag (captial O)
Set the compiler's optimization level.

Syntax: $ gcc -Olevel [options] [source file] [objectfil
[-o outputfile]
Run: $ gcc -O myfile.c -o myfile
$ .myfile
program run
$

(ii) gcc -s option
Remove all symbol table and relocation in
from the executable.
Appearently there is also a linker option -s
removes all symbol table and relocation in
from the executable

(iii) gcc -c option
Compilers source files without linking
Syntax: gcc -c [option] [source file]
Run: $ gcc -c myfile.c
→ This compilation generated myfile.o

(iv) gcc -save-temps function

→ The option can do all the work done in output : at all the stages of compilation is stored in the current directory. This option produces the executable.

Ex

$ gcc -save-temps main.c

$ ls

a.out main.c main.i main.o main.s

→ we can see all the intermediate files as well as the final executable.

(v) gcc -v option

used to provide verbose information on all the steps gcc takes while compilling a source file.

$ gcc -v main.c -o main

using built-in specs

COLLECT_GCC = gcc

COLLECT_LTO_WRAPPER = /usr/lib/gcc/i68...

Target: i686-linux-gnu

Configured with: ../src/configure -v ...

Thread model: posix

(3) LLVM:

(1) understand and discuss this key word.
OpenCL, OpenGL, OpenMP, clang, LLVM, Bison.

OpenCL: OpenCL (Open Computing Language) is a low-level API for heterogeneous Computing that runs on CUDA-powered GPUs. Using OpenCL API, developers can launch Compute kernels written using a limited subset of the C Programming language on a GPU.

OpenGL: OpenGL is a software interface to graphics hardware. This interface consists of more than 700 distinct commands that you see use to specify the objects and operations needed to produce interactive three-dimensional applications.

→ OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different platforms. To achieve these goals, it has no commands for performing windowing tasks or obtaining user input, instead including in OpenGL:

OpenMP : OpenMP (Open Multi-Processing) is an application programming interface (API) that supports Multi-Program shared-memory multi-Processing Programming in C, C++, and fortran on many platforms, instruction-set architectures and operating Systems including Solaris, AIX, FreeBSD, HP-UX, Linux, MacOS, and Windows.

- OpenMP uses a portable, scalable model that gives programmers a simple and flexible interface for developing parallel applications for platforms ranging from the standard desktop Computer to the Super Computer.
- OpenMP is an implementation of Multi-threading, a Method of parallezing where by a primary thread forks a specified number of sub-threads and the system devides a task among them

clang : clang is a Compiler front end for C, C++, Objective-C Programming languages as well as openMP, Opencl, Cuda framework

- clang uses LLVM Compiler as its backend
- clang is built be drop in replacement for gcc.

LLVM : LLVM stands for low level machine lang
→ LLVM is a compiler and a toolkit for
building compilers, which are program
Convert instructions into a form that
be read and executed by a compute
→ LLVM helps build new computer langu
and improve existing language. It
automates many of the difficult and
unpleasant tasks involved in langu
creation, such as porting the outputt
code to multiple platforms and archi

Bison : Bison is a general-purpose parser gene
that converts a grammer description f
an LALR(1) context-free-grammer in
C program to parse that grammer. T
Bison parser is a bottom-up parser.
It tries, by shifts and reductions, t
reduce the entire input down to a
single grouping whose symbol is th
grammer's start-symbol.

② what is LLVM, explain its feature?

- It is a set of compiler and tool chain technology that can be used to develop a front end for any programming language and backend for any instruction set Arch.

features:

- fast compilers and low memory use
- Expressive Diagnosites
- GCC Compatibility
- fronted for C, C++, etc...
- use LLVM, 'Apache', License
- A stable implementation of LLVM inst^n set
- library based architecture.

③ What languages are using LLVM?

- LLVM currently supports compiling of C, C++, Delhi, fortran, Haskel, Jolio, Rust and Swift using various frontend.

④ What are the components of LLVM?

↳ 1) frontends

- The modification generally involve a simple to LLVM IR step so that LLVM optimizes.

2) Intermediate Representation

- It support these equivalent form of IR
  A human - readable assembly.

(3) Backend :

- The LLVM code Subproject is LLVM's fra for translating b/w textual forms & machine code.

4) Linkers
5) Polly
6) Debuggers
7) C++ standard library
8) C standard library.

(5) What is clang? What are its features?

→ clang is a compiler frontend for C, C++, e program lang as well as OpenMP, OpenCL, u frameworks.

features:

- faster than gcc compilation
- uses less memory
- generates errors & warning messages more useful.
- support diverse clients
- Support wide range of language exten

- find and explain features of toy language like 'Bhai lang' ?

→ toy lang refers to any computer programming that is not considered to be suitable or capable building general purpose application.

→ It may also be termed as esoteric programming

- This lang. is build on Javascript.

- It has some terminology like:
  - Entry & Exit - point
  - variables
  - Datatypes
  - Outputs
  - loops
  - Errors

Example : Bhailang

```
hi bhai
    bol bhai "Hello word";
    bhai ye bhai a = 3;
    Jab tak bhai (b<5)
    {
     bol bhai b;
     agar bhai (b==a)
       {
        bol bhai "b is equal to a";
       }
```

```
nahi to bhai (b==0)
{
    bol bhai "b is equal to zero";
    b+=1;
}
bye bhai
}
```

(4) Explain how "JFLAP" can be used to design lang.

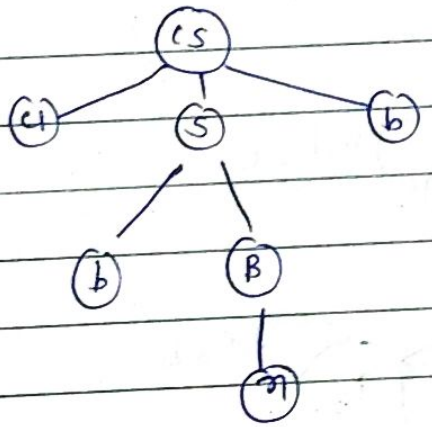
- JFLAP is software for programming with f lang. topics.
- It is a package of graphical tools which be used as can aid in learning the ba concept of formal language & Automata

→ Regular lang - create
- DFA
- NFA
- Grammer
- expression

- Regular lang - conversions
  - NFA → DFA → minimal DFA
  - NFA ↔ Reg. Expression
  - NFA ↔ Reg. grammer

→ Contex free lang - Create
- Push down Automation
- CFG

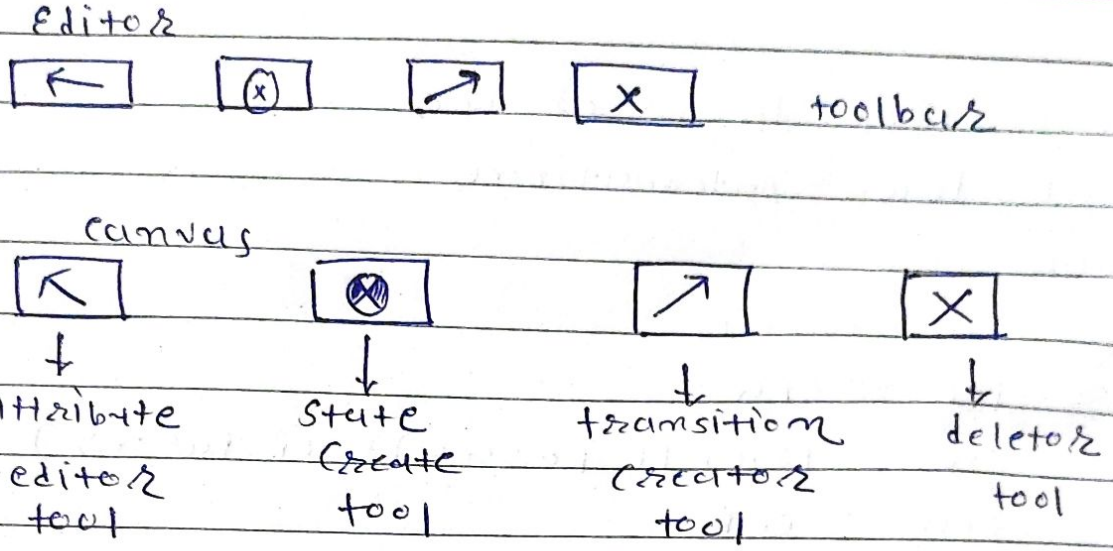
→ CFL - transform
PDA → CFG
CFG → PDA (LL Passes) (SLR Passer)
CFG → CNF
CFG → Brote force Passer



- Recursively Enumerable language
- TM | (1-type)
- Multitape TM
- building blocks TM
- Unrestricted grammer

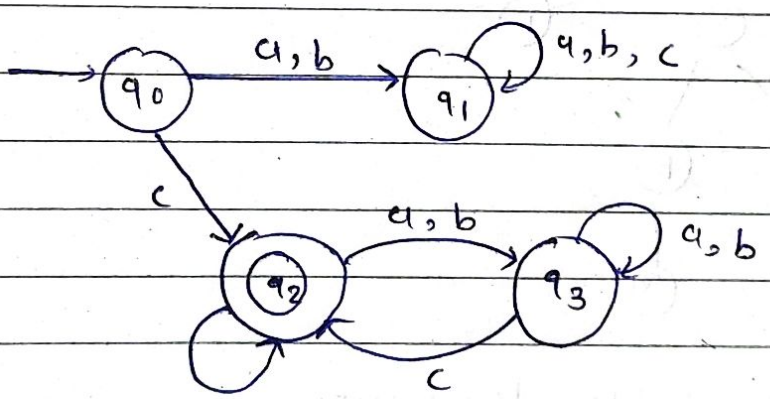
- L-systems
- Create L - systems

- Construct & Run
JFLAP
file Input test Convert help.



→ After Constructing below.

DFA :



→ Check your DFA to ensure that it accepts Reject the Starting your previously identi

Experiment : 2

Aim : Study FLEX TOOLS.

*Study flex manual and try to find answer for following questions :

1) what is flex ?

→ FLEX (fast lexical Analyzer generator) is a tool / computer program for generating lexical analyzers written by vern paxson in C.

→ It is used together GNU Bison parser generator or Berkeley Yacc parser generator.

2) what are characteristics of flex ?

→ Main characteristics are :

- for generating Scanners
- flex is a free and open-source s/w alternative to lex.
- flex reads given input files for a description of scanner to generate.
- The Description is in form of pairs of Regular expression & C code called Rules.

What is format of flex?

→ (i) defination section

```
%{
    // definations
%}
```

(ii) Rules Sections

```
%%
    pattern action
%%
```

(iii) user code section

```
%{
    // definant
%}
%% rules
%%

    user code secomtion.
```

* Explain steps to use flex.

↪ step1 : lex filenamed .l or .lex

step2 : gcc lex.yy.c

step3 : ./a.out

step4 : provide input to program if required.

Is following FLEX input program valid ?
if 'Yes' how is the resultant Scanner ? y.y.

→ Yes, FLEX input program is valid.

%%% → generates rule section according to pattern actions.

② Generate Scanner which identifies token 'username and displays current logged in username in response.

```
$who
$w
$users
```

③ Generate Scanner using lex, which simply counts number of lines & number of characters in input & display it when EOF encounted in input.

→
```
%{
    int no_of_lines = 0;
    int no_of_chars = 0;
%}
```

```
%%
\n      ++ no_of_lines;
.       ++ no_of_chars;
Eof     return 0;
%%

int yywrap(){}
int main(int argc, char * argv)
{
    yylex();
    printf("number of lines = %d,
            number of chars = %d \n",
            no_of_lines, no_of_chars);

    return 0;

}
```

Experiment = 3

Aim : Using flex tool generate scanner for a language.

Procedure :

Generate scanner using lex, which can identify following language - "D", and display appropriate message on recognizing them.

Rules of "D" language :

(a) keywords :
if, for, while, do, exit, else, case, switch, until.

(b) identifier :
String with alphabet (big or small) and followed by one or more alphabets or numbers.

(c) Numbers :
① float number :
having one '.' and valid format are
→ .Num or Num.Num or Num.0

② integer number : Simple Numbers, starting with digit followed by zero or more digits.

(d) Single line comments string with "**"

(e) white space characters are - blank & tabs

(f) Punctuations are - [,], {, }, (,) and comma

(g) operators :=, <>, ==, != , <=, >= (relational)
+, -, *, / (mathematical)

(h) string constant in 'note - no enclosing quote allowed in language.

example

```
%%

if|for|then|while|do|exit|else|case|switch|until
        {printf("%s found keyword \n", yytext);}

[0-9]* {printf ("%s its a integer number \n",
                yytext);}

[0-9]*[0-9][.][0-9]+ {printf ("%s its a float
                number \n", yytext);}

([a-zA-Z])* {printf ("%s its a identifier \n", yytext);}
[a-zA-Z][([a-zA-Z0-9])*

:=|==|>=|>|>=|<|<=|<>|!=
        {printf ("%s its relational operator", yytext);}

[+|-|*|/] {printf ("%s its arithmetic operator", yytext);}

. {printf (" not found\n");}
\n return 0;
%%
```

```
int yywrap(){}

int main()
{
printf("start\n");
yylex();
printf("end\n");

}
```

RECORD OF WORK DONE

NO.	EXPERIMENT	PAGE NO.	DATE OF EXPERIMENT	DATE OF REPORT	SIGNATURE
1)	Study the process of translation		12/12/22		
2)	Study flex tool		19/12/22		
3)	Using flex		26/12/22		