JSS MAHAVIDYAPEETHA
JSS SCIENCE AND TECHNOLOGY UNIVERSITY
SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING

JSS Technical Institutions Campus, Mysuru – 570 006

A seminar report on

"Lex and Yacc program which accept strings that starts and ends with Zero or One"
20IS530

Bachelor of Engineering
in
INFORMATION SCIENCE AND ENGG

Submitted to

Shruti N
Assistant professor
IS department

by,

Suhas M S- 01JST20IS045

DECEMBER 2022

Aim: LEX and YACC program which accept strings that starts and ends with Zero or One.

Source code:

program1.l

```
%{
/* Definition section */
extern int yylval;
%}

/* Rule Section */
%%

0 {yylval = 0; return ZERO;}

1 {yylval = 1; return ONE;}

.|\n {yylval = 2; return 0;}

%%
```

program1.y

```
%{
/* Definition section */
#include<stdio.h>
#include <stdlib.h>
void yyerror(const char *str)
{
printf("\n Sequence Rejected\n");
```

```
    }

%}

%token ZERO ONE

/* Rule Section */
%%

r : s {printf("\nSequence Accepted\n\n");}
;

s : n
| ZERO a
| ONE b
;

a : n a
| ZERO
;

b : n b
| ONE
;

n : ZERO
| ONE
;

%%

#include"lex.yy.c"
//driver code
int main()
{
        printf("\nEnter Sequence of Zeros and Ones : ");
```

```
        yyparse();
        printf("\n");
        return 0;
}
```

Output:

```
        user@user:~/syssoft$ lex program1.l
        user@user:~/syssoft$ yacc  program1.y
        user@user:~/syssoft$ gcc y.tab.c -ll
        user@user:~/syssoft$ ./a.out

        Enter Sequence of Zeros and Ones : 00100

        Sequence Accepted

         user@user:~/syssoft$ ./a.out

        Enter Sequence of Zeros and Ones : 10101

        Sequence Accepted

        user@user:~/syssoft$ ./a.out

        Enter Sequence of Zeros and Ones : 10100

        Sequence Rejected

        user@user:~/syssoft$ ./a.out

        Enter Sequence of Zeros and Ones : 00101

        Sequence Rejected
```

Lex and Yacc Program which accept strings that starts and ends with Zero or One
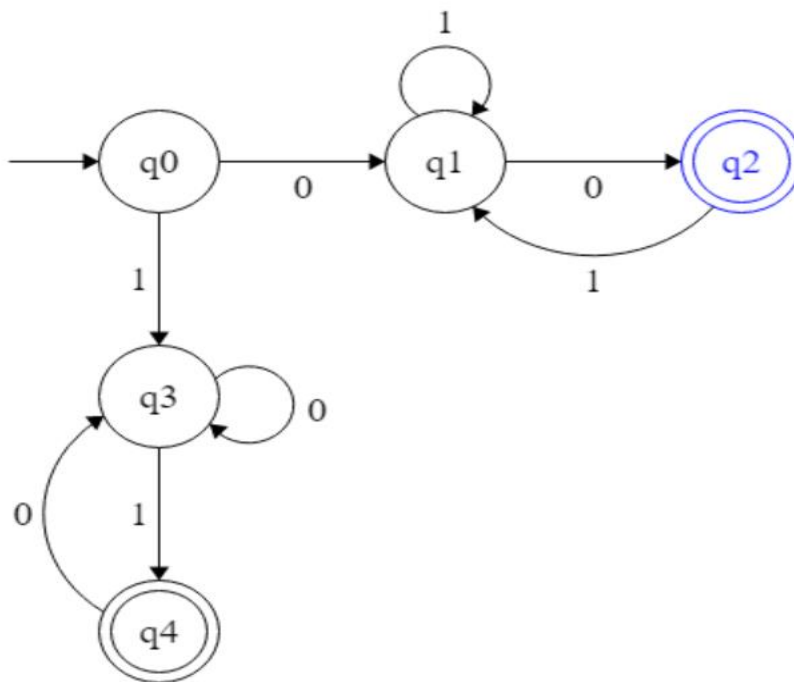
```
suhas@suhas-Inspiron-14-5410:~$ lex program1.l
suhas@suhas-Inspiron-14-5410:~$ yacc program1.y
suhas@suhas-Inspiron-14-5410:~$ gcc y.tab.c -ll
suhas@suhas-Inspiron-14-5410:~$ ./a.out

Enter Sequence of Zeros and Ones : 00100

Sequence Accepted


suhas@suhas-Inspiron-14-5410:~$
```

Explanation:

- Initially the lex program is created using the program1.l to generate the tokens for the program.
- The header "y.tab.h" is included in the definition section of the lex program to link the program with the corresponding yacc file.
- In the rules section, anything excluding numbers is ignored and then only numbers between 0 or 1 with atleast one occurrence is returned to the yacc code as a token.
- If the expression is invalid yyerror() throws an error to the console.
- After the tokens are generated yywrap() function is called for yacc program.
- Once tokens are generated for the program yacc file is executed for the parse tree generation.
- The parse tree generated is in the tab.c file.
- The lex and yacc files produce lex.yy.c and y.tab.c files respectively.
- Once the files are generated they are compiled with the C compiler to produce the executable file i.e., ./a.out
- The files are compiled using the command gcc y.tab.c -ll .
- -ll specifies to OS to have the lex file address in the linker section.
- If any invalid expression is given then the program quits abnormally.

Conclusion:

- By the end of this project, I learnt Lex and Yacc are tools that are commonly used in the development of programming languages and compilers.
- Lex (short for Lexical Analyzer) is a program that takes a stream of characters as input and converts it into a stream of tokens, which are symbols that represent the elements of the language being analyzed.
- Yacc (short for Yet Another Compiler Compiler) is a tool that takes a stream of tokens as input and generates a parser that can

analyze the structure of the input and determine whether it is a valid program in the language being parsed.

- Together, Lex and Yacc can be used to develop compilers for programming languages and other tools that need to analyze and understand the structure of text input.
- They are often used in conjunction with other tools, such as a code generator or interpreter, to create a complete compiler or translation system.