Term Work

Compiler Design

(PCS-601)

2022-23

Submitted to:

Mr. Mukesh Kumar

(Associate Professor)

GEHU, DDN

Submitted by:

Name: Yogesh K. Bhatt

Section: H

Roll No.: 64

Semester: 06

Student ID: 20011189

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

ACKNOWLEDGEMENT

I would like to particularly thank my Compiler Design Lab Faculty Mr. Mukesh Kumar for his patience, support, and encouragement throughout the completion of this Term work.

At last, but not the least I greatly indebted to all other persons who directly or indirectly helped me during this course.

Yogesh Kumar Bhatt
University. Roll No.- 2018886
B. Tech CSE-H-VI Sem
Session: 2022-23
GEHU, Dehradun

INDEX

13.	Design YACC/LEX code to evaluate arithmetic expression involving operators +, -, * and / without operator precedence grammar & with operator precedence grammar.			
14.	Design YACC/LEX code that translates infix expression to postfix expression.			
15.	Design Desk Calculator using YACC/LEX Code.			

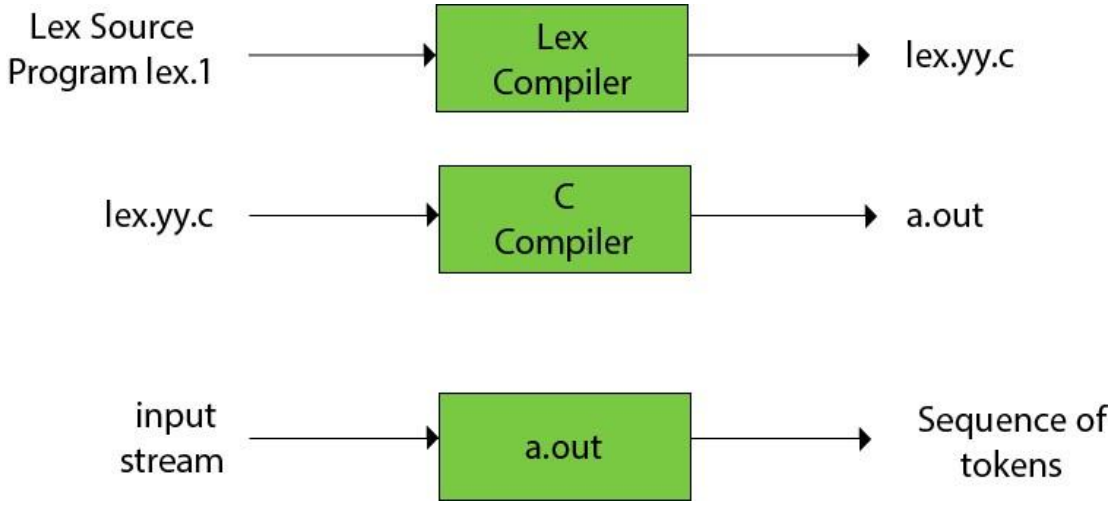
Study of Lex and Yacc

Lex:

- Lex is a tool which converts the source code into stream of tokens.
- It is also called scanner/tokenizer.

Functions of Lex:

- ✓ Firstly lexical analyzer creates a program lex.l in the Lex language. Then Lex compiler runs the lex.1 program and produces a C program lex.yy.c.
- ✓ Finally C compiler runs the lex.yy.c program and produces an object program a.out.
- ✓ a.out is lexical analyzer that transforms an input stream into a sequence of tokens.



Lex Program Format:

%{ Definition/Declaration Section

//header files, definitions, variables

%}

%%

Rule Section. //Regular expressions and corresponding actions.

%%
{Auxiliary Functions}

YACC:

- YACC stands for **Yet Another Compiler Compiler**.
- YACC provides a tool to produce a parser for a given grammar.
- The input of YACC is the rule or grammar and the output is a C program.
- YACC is a program designed to compile a LALR (1) grammar.
- It is used to generate the parse tree.

Functions:

File.y →[yacc tool/compiler]→y.tab.c

y.tab.c→[c compiler]→a.out tokens→[a.out]→[parse

tree with the help of the grammar]

Yacc Program Format:

The declarations section consists of two parts: (i) C declarations and (ii) YACC declarations .

```
%{

    /*Beginning of C declarations*/

   /*End of C declarations*/

%}

/*Beginning of YACC declarations  */

 /*End of YACC declarations */

/* End of Declarations Part */
%%

/* Rules Section begins here */

 --------------------

/* Rules Section ends here */
```

%%

{

Auxiliary functions

}

Program No. 1

Ques 1. Design a LEX Code to count the number of lines, space, tab-meta character and rest of characters in a given Input pattern.

Source Code:

```
%{
#include <stdio.h>
int l=0, c=0, s=0, t=0;
%}

%%

[\n] {l++;c++;}
[\t] {t++;c++;}
[ ] {s++;c++;}
[^\t\n ] {c++;}


%%

int yywrap() {
        return 1;
}
int main() {
        printf("Enter the input string: ");
        yylex();
        printf("\nLines=%d\nCharacters=%d\nSpaces=%d\nTab=%d\n", l, c, s, t);
        return 0;
    }
```

Output

```
D:\VS Code\CompilerDesign>a.exe
Enter the input string: Hello World     Again.
Testing.
^Z

Lines=2
Characters=28
Spaces=1
Tab=1
```

Program No. 2

Ques 2. Design a LEX Code to identify and print valid Identifier of C/C++ in given Input pattern.

Source Code:

```
%{
        #include <stdio.h>
%}

%%

[a-zA-Z_][a-zA-Z0-9_]* {printf("Valid Identifier");}
.* {printf("Not a Valid Identifier");}

%%

        int yywrap() {
                return 1;
        }
        int main() {
                printf("\nEnter the input: ");
                 yylex();
                return 0;
        }
```

Output

```
D:\VS Code\CompilerDesign>a.exe

Enter the input: abc
Valid Identifier


count
Valid Identifier


1count
Not a Valid Identifier
```

Program No. 3

Ques 3. Design a LEX Code to identify and print integer and float value in given Input pattern.

Source Code:

```
%{
        #include <stdio.h>
%}

%%

[0-9]+"."[0-9]+ {printf("\nDecimal Number\n");}
[0-9]+ {printf("\nInteger Number\n");}

%%

        int yywrap()
        {
        return 1;
   }
        int main()
        {
                yylex();
                return 0;
        }
```

Output

```
D:\VS Code\CompilerDesign>a.exe
66

Integer Number

66.6

Decimal Number
```

Program No. 4

Ques 4. Design a LEX Code for Tokenizing (Identify and print OPERATORS, SEPERATORS, KEYWORDS, IDENTIFERS).

Source Code:

```
%{
        #include <stdio.h>
%}

%%

[+ - = * /] {printf("Operator\n");}
[{}()] {printf("Separator\n");}
int|float|if|else|while|bool|for|do|double|char|printf|scanf|default|auto|goto|break|continue|case|
switch|enum|extern|inline|long|short|return|sizeof|signed|static|unsigned|typedef|union|void
{printf("Keyword\n");}
[a-z A-Z][a-z A-Z 0-9]* {printf("Identifier\n");}

%%

        int yywrap() {
                return 1;
        }
        int main() {
                yylex();
                return 0;
        }
```

Output

```
D:\VS Code\CompilerDesign>a.exe
+
Operator

{
Separator

count
Identifier

int
Keyword
```

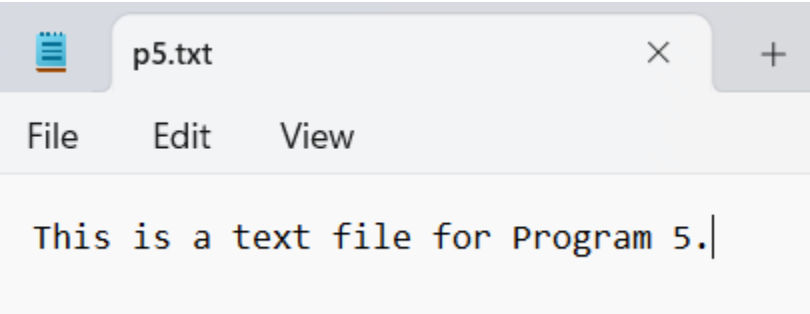
Program No. 5

Ques 5. Design a LEX Code to count and print the number of total characters, words, white spaces in given 'Input.txt' file.

Source Code:

```
%{
	#include <stdio.h>
	int tWord=0, tChar=0, tSpace=0;
%}

%%

[^\n \t] {tChar++;}
" " {tWord++; tSpace++;}
[\n \t] {tWord++;}

%%

	int yywrap() {
		return 1;
	}
	int main() {
		yyin=fopen("p5.txt", "r");
		yylex();
		printf("\nTotal Word = %d, Total Character = %d, Total Space = %d\n",
tWord, tChar, tSpace);
		return 0;
	}
```

Output




```
D:\VS Code\CompilerDesign>a.exe

Total Word = 7, Total Character = 27, Total Space = 7
```

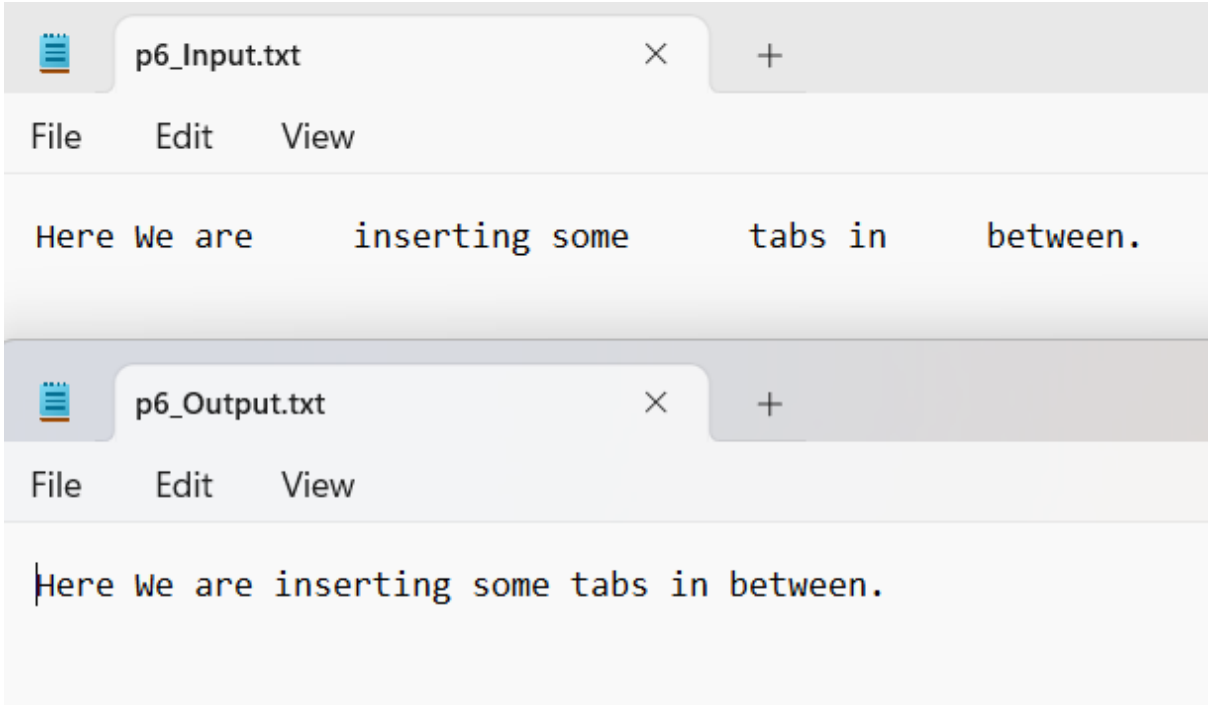
Program No. 6

Ques 6. Design a LEX Code to replace white spaces of 'Input.txt' file by a single blank character into 'Output.txt' file.

Source Code:

```
%{
	#include <stdio.h>
%}

%%

[\t " "]+ {fprintf(yyout, " ");}

%%

	int yywrap() {
		return 1;
	}
	int main() {
		yyin = fopen("p6_Input.txt", "r");
		yyout = fopen("p6_Output.txt", "w");
		yylex();
		return 0;
	}
```

Output:


p6_Input.txt
File Edit View
Here We are inserting some tabs in between.
p6_Output.txt
File Edit View
Here We are inserting some tabs in between.

Program No. 7

Ques 7. Design a LEX Code to remove the comments from any C Program given at run-time and store into 'out.c' file.

Source Code:

```
%{
	#include <stdio.h>
%}

%%

\\(.*) {};
\\*(.*\n)*.*\*\/ {};

%%

	int yywrap() {
		return 1;
	}
	int main() {
		yyin = fopen("p7_Input.c", "r");
		yyout = fopen("p7_Output.c", "w");
		yylex();
		return 0;
	}
```

Output:

p7_Input.c

File Edit View

```c
#include <stdio.h>
int main() {
	//This is a single line comment
	int i=3;
	while(i>3) {
		printf("%d", i);
	}
	/*
	 * Multi
	 * Line
	 * Comment
	*/
	return 0;
}
```

p7_Output.c

File Edit View

```c
#include <stdio.h>
int main() {

	int i=3;
	while(i>3) {
		printf("%d", i);
	}

	return 0;
}
```

Program No. 8

Ques 8. Design a LEX Code to extract all html tags in the given HTML file at run time and store into Text file given at run time.

Source Code:

```
%{
#include <stdio.h>
%}

%%

"<"[^>]*> {fprintf(yyout,"%s\n",yytext);}
. {};

%%

int yywrap() {
	return 1;
}
int main() {
	yyin = fopen("p8_htmlIn.html", "r");
	yyout = fopen("p8_htmlOut.txt", "w");
	yylex();
	return 0;
}
```

Output:

p8_htmlIn.html

File Edit View

```
<!doctype html>
<html>
<head>
<title>Title</title>
</head>
<body>
This is body
</body>
</html>
```

p8_htmlOut.txt

File Edit View

```
<!doctype html>

<html>

<head>

<title>
</title>

</head>

<body>


</body>

</html>
```