# BCSE307P

# Compiler Design Lab

# Lab Assignment 2


VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI


## Name – Ishan Kapoor

## Registration Number – 21BCE5882

## Submitted to – Prof. S. Srisakthi

1. A Lex program to take an input and count the number of vowels and consonants in it.

**Code:**

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

int vowelCount = 0;
int consonantCount = 0;
%}

%%
[aAeEiIoOuU]     { vowelCount++; }
[a-zA-Z]         { consonantCount++; }
\n {return 0;}
.                { /* Ignore other characters */ }
%%

int main() {
    yylex();
    printf("Vowel count: %d\n", vowelCount);
    printf("Consonant count: %d\n", consonantCount);
    return 0;
}
int yywrap(){
return 1;
}
```

**Output:**

```
ishan@ishan-VirtualBox:~/Desktop/Compiler$ lex vowels_consonants_counter.l
ishan@ishan-VirtualBox:~/Desktop/Compiler$ gcc lex.yy.c
ishan@ishan-VirtualBox:~/Desktop/Compiler$ ./a.out
Ishan
Vowel count: 2
Consonant count: 3
ishan@ishan-VirtualBox:~/Desktop/Compiler$ gcc lex.yy.c
ishan@ishan-VirtualBox:~/Desktop/Compiler$ ./a.out
Kapoor
Vowel count: 3
Consonant count: 3
```

2. A lex program for lexical analyser in C.

## Code:

- Lex Code

```
%%

#.*                                                                     {printf("Pre-processor directive: %s\n\n",yytext);}
['']|,|;(|)|{|}|.|_]                                                    {printf("Delimiter: %s\n",yytext);}
[[]]                                                                    {printf("Delimiter: %s\n",yytext);}
"<="|">="|"++"|"!="|"=="|"<"|">"                                        {printf("Operator: %s\n",yytext);}
"+"|"-"|"*"|"/"                                                         {printf("Operator: %s\n",yytext);}
"and"|"or"|"not"|"nand"|"xor"|"nor"|"xnor"                              {printf("Operator: %s\n",yytext);}
"&"|"|"|"!"|"^"                                                         {printf("Operator: %s\n",yytext);}
"="                                                                     {printf("Operator: %s\n",yytext);}
("int")|("if")|("else")|("while")|("do")|("break")|("continue")|("double")|("float")|("return")|("EOF") {printf("Keyword: %s\n",yytext);}
("return")|("char")|("case")|("sizeof")|("long")|("short")|("typedef")|("switch")|("unsigned")|("void")|("static")|("struct")|("goto")
  {printf("Keyword:%s\n",yytext);}
[a-zA-Z_$][a-zA-Z0-9_$]*                                                {printf("Identifier: %s\n",yytext);}
[-]?[0-9]+"."[0-9]+                                                     {printf("Constant : %s\n", yytext);}
[-]?[0-9]+                                                              {printf("Constant: %s\n",yytext);}
["][^"\\\n]*["]                                                         {printf("String constant:%s\n",yytext);}
[ ]                                                                     {;}
[\t\n]+
.                                                                       {printf("Not recognized: %s\n",yytext);}

%%

#include <stdio.h>

extern int yylex();
extern char* yytext;

int main(){
    yylex();
        return 0;

}

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

- Text File

```
#include <stdio.h>

int main(){
        int n, fact;
        scanf("%d",&n);
        int i=1;
        while(i<=n){
                fact = fact*i;
                i++;
        }
        printf("%d", fact);
}
```

# Output:

```
ishan@ishan-VirtualBox:~/Desktop/Compiler$ lex lexical_analyzer.l
ishan@ishan-VirtualBox:~/Desktop/Compiler$ gcc lex.yy.c
ishan@ishan-VirtualBox:~/Desktop/Compiler$ ./a.out<Factorial.c
```

```
Pre-processor directive: #include <stdio.h>

Keyword: int
Identifier: main
Delimiter: (
Delimiter: )
Delimiter: {
Keyword: int
Identifier: n
Delimiter: ,
Identifier: fact
Delimiter: ;
Identifier: scanf
Delimiter: (
String constant:"%d"
Delimiter: ,
Operator: &
Identifier: n
Delimiter: )
Delimiter: ;
Keyword: int
Identifier: i
Operator: =
Constant: 1
Delimiter: ;
Keyword: while
Delimiter: (
Identifier: i
Operator: <=
Identifier: n
Delimiter: )
Delimiter: {
Identifier: fact
Operator: =
Identifier: fact
```

```
Operator: *
Identifier: i
Delimiter: ;
Identifier: i
Operator: ++
Delimiter: ;
Delimiter: }
Identifier: printf
Delimiter: (
String constant:"%d"
Delimiter: ,
Identifier: fact
Delimiter: )
Delimiter: ;
Delimiter: }
```