Experiment No 5

Aim : Study of Lexical analyzer tool -Flex/ Lex

Leraning Objective: Recognise lexical pattern from given input file

Code And Output

Number of Vowels

```
%{
#include<stdio.h>
int vows=0;
%}
%%
[aeiouAEIOU] {vows++;}
%%
int yywrap()
{
return 1;
}
int main(){
printf("Enter the string: ");
yylex();
printf("The number of vowels are: %d", vows);
}
```

```
universe@acer11:~$ lex 9176_SPCC5.l
universe@acer11:~$ cc lex.yy.c
universe@acer11:~$ ./a.out
Enter the string: hello
hll
The number of vowels are: 2universe@acer11:~$
```

Number of Vowels and Consonants

```
%{
#include <stdio.h>
int vow = 0;
int consonantst = 0;
%}
%%
[aeiouAEIOU] {vow++;}
[a-zA-Z] {consonantst++;}
%%
int yywrap()
{
return 1;
}
void main()
{
```

```
printf("Enter string\n");
yylex();
printf("Number of Vowels: %d\n", vow);
printf("Number of consonantst: %d\n", consonantst);
}
```

```
universe@acer11:~$ lex 9176_SPCC5.l
universe@acer11:~$ cc lex.yy.c
universe@acer11:~$ ./a.out
Enter string
hello

Number of Vowels: 2
Number of consonantst: 3
universe@acer11:~$
```

Number of Positive Number, Negative Number, Positive Decimal and Negative Decimals

```
%{
#include <stdio.h>
int posnumber = 0;
int negnumber = 0;
int posdecimal = 0;
int negdecimal = 0;
%}
%%
[-][0-9]+ {negnumber++;}
[+]?[0-9]+ {posnumber++;}
[-][0-9]+\.[0-9]+ {negdecimal++;}
[+]?[0-9]*\.[0-9]+ {posdecimal++;}
%%
int yywrap()
{
return 1;
}
void main()
{
printf("Enter Number\n");
yylex();
printf("Number of Positive Number: %d\n", posnumber);
printf("Number of Negitive Number: %d\n", negnumber);
printf("Number of Positive Decimal: %d\n", posdecimal);
printf("Number of Negitive Decimal: %d\n", negdecimal);
}
```

```
universe@acer11:~$ lex 9176_SPCC5.l
universe@acer11:~$ cc lex.yy.c
universe@acer11:~$ ./a.out
Enter Number
30 -5 20 -1.4 0.2 .4

Number of Positive Number: 2
Number of Negitive Number: 1
Number of Positive Decimal: 2
Number of Negitive Decimal: 1
universe@acer11:~$
```

Number of Words, Spaces, End of Line and Characters

```
%{
#include <stdio.h>
int words = 0;
int space = 0;
int endoflines = 0;
int characters = 0;
%}
%%
([a-zA-Z0-9]+[a-zA-Z0-9])* {words++;}
([ \t]) {space++;}
([\n]) {endoflines++;}
. {characters++;}
%%
int yywrap()
{
return 1;
}
void main()
{
printf("Enter String\n");
yylex();
printf("Number of words: %d\n", words);
printf("Number of spaces: %d\n", space);
printf("Number of end of line: %d\n", endoflines);
printf("Number of characters: %d\n", characters);
}
```

```
universe@acer11:~$ lex 9176_SPCC5a.l
universe@acer11:~$ cc lex.yy.c
universe@acer11:~$ ./a.out
Enter String
hello world a
Number of words: 2
Number of spaces: 2
Number of end of line: 1
Number of characters: 1
universe@acer11:~$
```

Number of Operators and Operands

```
%{
#include <stdio.h>
int operator = 0;
int operand = 0;
%}
%%
[\+\-\=\/\%] {operator++;}
[a-zA-Z0-9]+ {operand++;}
%%
int yywrap()
{
return 1;
}
void main()
{
printf("Enter Expression\n");
yylex();
printf("Operator: %d\n", operator);
printf("Operand: %d\n", operand);
}
```

```
universe@acer11:~$ lex 9176_SPCC5.l
universe@acer11:~$ cc lex.yy.c
universe@acer11:~$ ./a.out
Enter Expression
a=b+c

Operator: 2
Operand: 3
universe@acer11:~$
```

Number of Keywords, Operators, Constants, Identifiers and Special Symbols

```
%{
#include <stdio.h>
int keywords = 0;
int operator = 0;
int constants = 0;
int special = 0;
int identifiers = 0;
%}
%%
int|float|char|return|for|while|do|if|else {keywords++;}
[\+\-\=\/] {operator++;}
[0-9]+ {constants++;}
[/%/&/;/$] {special++;}
[_||a-zA-Z][_a-zA-Z0-9]* {identifiers++;}
%%
int yywrap()
```

```
{
return 1;
}
void main()
{
printf("Enter Line of code\n");
yylex();
printf("Keywords: %d\n", keywords);
printf("Operators: %d\n", operator);
printf("constants: %d\n", constants);
printf("Identifier: %d\n", identifiers);
printf("Special Symbols: %d\n", special);
}
```

```
universe@acer11:~$ lex 9176_SPCC5.l
universe@acer11:~$ cc lex.yy.c
universe@acer11:~$ ./a.out
Enter Line of code
int bc=a-1;

Keywords: 1
Operators: 2
constants: 1
Identifier: 2
Special Symbols: 1
universe@acer11:~$
```

```
universe@acer11:~$ ./a.out
Enter Line of code
int _ab=c-d;

Keywords: 1
Operators: 2
constants: 0
Identifier: 3
Special Symbols: 1
universe@acer11:~$
```

Conclusion: Thus we learnt about lex and implemented it.

Postlab:

1. **Write the structure of Lex**
2. **Write the structure of Yacc**