Objectives:

1/ To know about lexical analysis.

2/ To learn how to use flex in lexical analysis.

Introduction:

Lexical analysis means dividing the input into meaningful units. For C program the units are variables, constants, keywords, operators etc. These units are also called tokens.

Flex is a tool for lexical analyzer generator. Flex source is a table of regular expressions and corresponding program fragment. It generates lex.yy.c which defines a routine yylex().

In this Lab we solved some problems using flex which will be dicussed in this report.

Theory:

Flex specifications –

```
{definitions}
%%
{rules}
%%
{user subroutine}
```

e.g.

```
%%
. {printf("character");}
%%
int yywrap()
{
  return 1;
}
main()
{
  yylex();
} printf("Hello");
```

Flex regular expressions:

1/ \n → newline
2/ \t → tab
3/ \0 → null character
4/ [abc] → matches either ~~a or~~ a, b or c
5/ [^abc] → Matches any character except a,b or c
6/ [a-z] → Matches any character betn a to z.
7/ [^a-z] → except a to z
8/ [a-zA-Z] → betn a-z or A to z
9/ . → Any single character
10/ a|b → either a or b
11/ \s → any space, tab or newline
12/ \S → anything except space, tab or newline.

13/ \d → any digit ; [0-9]
14/ a? → zero or one of a
15/ a* → zero or more of a
16/ a+ → One or more of a
17/ a{3} → exactly 3 of a
18/ a{3,} → 3 or more of a
19/ a{3,6} → beth 3 and 6 of a
20/ ^ → start of string
21/ $ → End of string
22/ (a) → Grouping

Discussion:

To create lexical analyzer we ~~need~~ ~~to~~ define regular expression carefully.

Detect ~~and~~ positive and negative integer -

```
%%
[+]?[0-9]+  {printf ("Positive\n");}
[-]?[0-9]+  {printf ("negative\n");}
%%

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

Discussion:

To create lexical analyzer we need to define regular expression carefully. First we need to think logically correct regex then we need to implement it using flex. Flex program extension is .l and need to run in command line. Flex is very flexible in defining complex lexical ~~an~~ rules using ~~ex~~ regex.

Conclusion:

Lexical analysis is a critical phase in compiler construction. and tools like flex simplify the process of implementing efficient lexical analyzer. Flex is a valuable tool for developers and compiler designers, streamlining the often intricate task of lexical analysis.