Exercise – 6: Use the Lex tool for the separation of token from a text.

Aim: To Use LEX tool to implement a lexical analyzer

Theory:

The compilation process goes through a sequence of steps in a predef*ined manne*r. From the given lexical and syntactical specifications, the scanning and parsing operations can be carried out as shown in exerices from 2-5. However, the process is repeatative in nature and hence it is automated by developing a tool. Lex tool is used to scan through the text and separate the tokens depending on the lexical speicication in the file. If this Lex tool is used in the compilation process, the C program embedded into the action part of the lexical specification file returns the tokens to the parser. Lex tool can be used for other purposes such text formatting, word separation, frequency count of characters, words and sentences etc. The output of the Lex tool is lex.yy.c, which consists of the scanning routines.

Steps:

1. Specify the lexical patterns in lex specification file (file name extension is .l).
2. Compile the lexical specification file using the tool Lex, which produces a file lex.yy.c.
3. Include lex.yy.c in a C file and compile using C compiler producing an executable version of the scanner program.
4. Call the scanner program in C program using the function yylex( ).

Modules:

- Auxiliary definitions (if any)
- Lexical pattern
- The actions for displaying the tokens

- The actions for returning the token to the parser

A sample lexical specification file as reproduced from exercise 3.23 is shown as follows with actions to display the type of operator:

```
%START BM
%%
[a-zA-Z][0-9a-zA-Z]*    {
                              BEGIN BM;
                              printf(“%s is an id\n”,yytext);
                        }
"+"                     {
                              BEGIN 0;
                              printf(“+ is a binary operator\n”);
                        }
"*"                     {
                              BEGIN 0;
                              printf(“* is a binary operator\n”);
                        }
[0-9]+                        {
                              BEGIN BM;
                              printf(“%s is a constant\n”);
                        }
<BM>“-”                 {
                              BEGIN 0;
                              printf(“- is a binary operator\n”);
                        }
“-”                     {
                              printf(“- is an unary operator\n”);
```

```
                    }
%%
```

The introduction to Lex tool with simple program is available in section 3.5.5 and the details of the usage of the Lex tool is given in **Chapter 6 and Appendix-A** for representing the entire C language. Refer example 3.24 for the count of the characters, words and lines in a file and example 3.25 for the replacing of the lower case letters by upper case letters.
