Practical 3: Lex Compiler for Lexical Analysis

A. Installation of Lex

Installing Lex on Windows:

1. **Download and install Cygwin**: Cygwin is a Linux-like environment for Windows that provides a command-line interface and a set of tools that are required to build and run Lex programs.

   a. Go to the Cygwin website at https://www.cygwin.com/
   b. Click on the "Install Cygwin" button to download the installer
   c. Run the installer and select the following packages: gcc-core, flex, and make.
   d. Follow the on-screen instructions to complete the installation.

2. **Download and install Lex**: Lex is a tool that generates lexical analyzers from regular expressions.

   a. Download the latest version of Lex from the GNU website at https://www.gnu.org/software/flex/
   b. Extract the contents of the downloaded archive to a directory of your choice.
   c. Open a Cygwin terminal and navigate to the directory where you extracted Lex.
   d. Run the following command to configure and build Lex:

   ```
   ./configure && make && make install
   ```

   e. Follow the on-screen instructions to complete the installation.

   Verify the installation: To verify that Lex is installed correctly, run the following command in the Cygwin terminal:

   ```
   flex –version
   ```

   Installing Lex on Linux-

   Install Lex using the package manager: Most Linux distributions come with a package manager that allows you to install software easily.

   a. Open a terminal window
   b. Run the following command to update the package manager:

   ```
   sudo apt-get update
   ```

   c. Run the following command to install Lex:

sudo apt-get install flex

3. Verify the installation: To verify that Lex is installed correctly, run the following command in the terminal:

   flex -- version

Example Program: Counting Words

The following program uses Lex to count the number of words in a text file. It assumes that a word is any sequence of one or more non-whitespace characters.

```
%{
#include <stdio.h>
int word_count = 0;
%}

%%
[a-zA-Z]+ { word_count++; }
%%

int main(int argc, char** argv) {
   yylex();
   printf("Word count: %d\n", word_count);
   return 0;
}
```

This program starts with a set of definitions and declarations between %{ and %}. In this case, we include the standard input/output library (stdio.h) and declare a variable word_count that will be used to store the number of words in the input file.

The program then defines the rules for recognizing words using regular expressions and actions using C code. In this case, we define a pattern for a word as any sequence of one or more alphabetic characters ([a-zA-Z]+). When a word is recognized, we increment the word_count variable.

Finally, the program includes the main function that initializes the lexical analyzer and calls the yylex() function to start scanning the input file. When the input is fully scanned, the main function prints the final word count.

Basic Steps to Execute the Program
Save the program in a file with a .l extension, for example, word_count.l.

Compile the program using the following command:

flex word_count.l

This generates a C source file named lex.yy.c.

Compile the C source file using a C compiler, such as gcc, with the following command:

gcc lex.yy.c -o word_count

This creates an executable file named word_count.

Run the program by passing the input file as a command-line argument:

./word_count input.txt

This will scan the input file input.txt and print the final word count.

That's it! You have now successfully written and executed a program using Lex to count the number of words in a text file.

Lexical Analyzer for a sample calculator

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

DIGIT [0-9]

%%
"+"     { printf("PLUS\n"); }
"-"     { printf("MINUS\n"); }
"*"     { printf("TIMES\n"); }
"/"     { printf("DIVIDE\n"); }
{DIGIT}+ { printf("NUMBER: %s\n", yytext); }
.       { printf("INVALID CHARACTER: %s\n", yytext); }
%%

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

Questions:

1. Write a sample lex program that builds a simple calculator that supports addition and subtraction of integers.