Lab Assignment 3- Lexical Analysis

Q1.

Q 1.1 Write a LEX/Flex program that recognizes binary strings containing even number of 0's.

Q 1.2 [Optional] Write a LEX/Flex program that recognizes binary strings containing even number of 0's and even number of 1's.

Q 1.3 Write a LEX/Flex program that recognizes binary strings whose integer equivalent is divisible by 3.

Q2. We had discussed about the lexical analyzer generator Lex/ Flex. Consider the example grammar for branching statements discussed in the class given below:

$$
\begin{array}{lcl}
\textit{stmt} & \rightarrow & \textbf{if } \textit{expr} \textbf{ then } \textit{stmt} \\
 & | & \textbf{if } \textit{expr} \textbf{ then } \textit{stmt} \textbf{ else } \textit{stmt} \\
 & | & \epsilon \\
\textit{expr} & \rightarrow & \textit{term} \textbf{ relop } \textit{term} \\
 & | & \textit{term} \\
\textit{term} & \rightarrow & \textbf{id} \\
 & | & \textbf{number}
\end{array}
$$

The patterns for the tokens in the language are described below:

$$
\begin{array}{rcl}
\textit{digit} & \rightarrow & \texttt{[0-9]} \\
\textit{digits} & \rightarrow & \textit{digit}^{+} \\
\textit{number} & \rightarrow & \textit{digits}\ (\texttt{.}\ \textit{digits})?\ (\ \texttt{E}\ \texttt{[+-]}?\ \textit{digits}\ )? \\
\textit{letter} & \rightarrow & \texttt{[A-Za-z]} \\
\textit{id} & \rightarrow & \textit{letter}\ (\ \textit{letter} \mid \textit{digit}\ )^{*} \\
\textit{if} & \rightarrow & \texttt{if} \\
\textit{then} & \rightarrow & \texttt{then} \\
\textit{else} & \rightarrow & \texttt{else} \\
\textit{relop} & \rightarrow & \texttt{<} \mid \texttt{>} \mid \texttt{<=} \mid \texttt{>=} \mid \texttt{=} \mid \texttt{<>}
\end{array}
$$

Q 2.1. Write a Lex/Flex program to describe the tokens of the above grammar, and generate a lexical analyzer using the Lex/Flex tool.

Q 2.2 Test the lexical analyzer with some input strings (You should show and explain the output of the lexical analyzer for the considered examples).

Q 2.3 [**OPTIONAL**] We had discussed about how to transform the patterns to transition diagrams by hand and how to implement a lexical analyzer. Implement a lexical analyzer for the considered problem without using the Lex/Flex tool.

Q 3 Construct a lexical analyzer for the following simple "C" like language using the Lex/Flex tool.

1. Data Type : integer (INT/int), floating point (FLOAT/float)
2. Condition constructs: **if**
3. Loop Constructs: for, while
4. Input / Output Constructs:
    a. read(x) - Read into variable x
    b. print(x) - Write variable x to output
5. Relational operators, assignment and arithmetic operators
6. Only function is **main()**, there is no other function.

You may test it using the below example:

Example:

```
main()
{
        INT i=0;
        INT sum=0;
        INT count;

        read(count);
        for(i=0;i<10;i++)
        {

                read(x);
                sum+=x;

        }

        print(sum);
}
```