**TEST PLAN**

**Implementation details:**

This project is an implementation of Lexical Analyzer using C programming which would consist of a driver function to take input from the user in form of character stream and further provide this to a lexer.

Another, function is created to perform the lexer tasks, which consists of conditional loops for comparison to generate the token types based on the symbol table. These symbol tables are written in the program using functions of type Boolean that would return TRUE/FALSE based on the presence of the tokens in the function defined. Finally, the output is generated where the tokens along with their symbol type is printed in the output console.

## Sample Specification:

In this section a complete specification for a simple lexical analyzer is developed. This analyzer will be required to recognize and return the components of arithmetic expressions, skipping white spaces. Specifically, the components to be recognized are:

1. Valid Identifiers
2. Invalid Identifiers
3. Integer and real numbers
4. Keywords (“int”, “float”, etc.)
5. Operator symbols ("+", "-", "\*", and "/")

## Actions:

Actions must be specified for each of the patterns to be recognized. To simplify this discussion, assume there exists a function called “LEX” that creates a token value from the information collected by the lexical analyzer.

The requirements for this function will be conditional statements such as “if”, “if-else” to compare the result generated from the function to the specified Boolean value. Here the character streams are first broken into simple character substrings and then are used for comparison by the created functions. The use of this would be clear once the programming is done.

 Finally, the Specification Test and Unit test should be conducted to check the behaviour of the written program against various test cases provided by the user. This would ensure the quality of the program, i.e. it would check if the code meets the requirements laid out in this document.