References:

- Compilers : Principles,Techniques and Tools by Alfred V.Aho, Monica S. Lam, Ravi Sethi and Jeffrey D.Ulman. https://www-2.dc.uba.ar/staff/becher/dragon.pdf
- Modern Compiler Implementation in C by Andrew W.Appel
- Modern Compiler Implementation in JAVA by Andrew W.Appel
- lex & yacc, 2nd Edition by Doug Brown, John Levine, Tony Mason
- Flex & Bison by John Levine
- http://dinosaur.compilertools.net/
- https://docs.oracle.com/cd/E19504-01/802-5880/6i9k05dgg/index.html
- https://docs.oracle.com/cd/E19504-01/802-5880/6i9k05dgt/index.html
- https://www.ibm.com/docs/en/zos/2.4.0?topic=lex-input-language
- https://silcnitc.github.io/lex.html
- https://web.stanford.edu/class/cs143/
- https://westes.github.io/flex/manual/
- https://www.gnu.org/software/bison/manual/html_node/index.html
- https://arcb.csc.ncsu.edu/~mueller/codeopt/codeopt00/y_man.pdf
- https://nxmnpg.lemoda.net/1/lex
- https://nptel.ac.in/courses/106104123

Project

Design a compiler for any programming language of your choice

Tasks:

- Create Compiler for your chosen programming language (You can create your own programming language using regional language as well)
    - **Lexical Analyzer**
        - **Regular Expressions for your language, Actions, Tokens**
        - **Handling of Errors**
    - Parser
        - Parse Tree
    - Intermediate Code Generation
    - Code Optimization
    - Target Code
- Write a sample source program for calculator in the language you developed and compile the program by your compiler
- Write test programs to check every statement of the compiler and show that it is working correctly.

Assignment 1:

1. Develop the *components* of your programming language.
2. Write regular expressions for each of them and draw the corresponding DFA
3. Write Lex code implementing the patterns and corresponding actions.
4. Write codes for handling errors during lexical analysis.
5. Compile the Lex code and create your own lexical analyzer (L).
6. Write small programs in the language you have developed.
7. Compile your programs and show that your lexical analyzer is creating correct tokens and handling errors correctly.