Compiler Mini Projects (In groups of 3):

00. Implement a shift reduce parser using C/C++.

01. Design a mini C compiler which will read a C program and check it for syntax errors.

02. Generate a lex compiler using lex programming. Implement the simple features.

03. Construct a mini C++ type compiler which is able to strictly identify only C++ code and report error on any C code which is acceptable in C++ in general (It separates C from C++, in other words).

04. Write a mini-Pascal compiler that reads simple Pascal programs. It should report the syntax errors. Check for conditional and loop statements efficiently.

05. Write a mini java compiler which detects simple java programs. It should report all the syntax errors in a given program properly.

06. Design a mini CJ (C-Java) compiler that can accept any C or java program and identify it. It should detect the syntax errors in each of these input programs. It will have less features as compared to individual C and Java compiler features.

07. Write a 'ParserPack' which can parse any statement and differentiate the ambiguity arising in the grammars. It should report all the ambiguous cases properly.

08. Design a Mini package to accept and compile C# and java programs. Both the kinds of programs should be checked for syntax errors. Handle the case if the codes of C# and java are mixed; report which code belongs to which language.

09. Generate to parser based on backtracking parser. It should identify the numeric data– integers, real numbers and complex numbers in proper format.

10. Write a 'XPack' interpreter/compiler which can accept an XML file as input and process it for syntax check. Define necessary rules and guidelines in a catalog for this pack.

11. Generate a mini compiler 'Easy' which reads a new programming language, which is near to natural language. Define the rules for it first in which include basic data types, elementary mathematical operations, conditional and iterative statements and function definitions. It should be different from existing C and similar languages. Prepare a manual including all these rules first of all.

12. Define a pseudocode convention for writing a program/algorithm and list them in a manual. Now write a tool for accepting the programs written in this pseudolanguage.

13. Write a compiling module for logical programming. It takes simple propositions as input and processes them. Use PROLOG programming as the base of this project.

14. Write a 'QCompiler' which will accept the SQL statements embedded in html. It should detect the syntax errors and suggest the necessary corrections. In order to reduce ambiguity, multiple suggessions should be provided if any.