

# Objectives:

**CPCS 302 COMPILER CONSTRUCTION**

**Term Project**

This project is about designing your programming language! So, you need to construct its **compiler**.

You will work as groups to build a compiler using JavaCC. The compiler should read your language code and display output that corresponds to the requirements in the three phases as follows:

Phase 1: Lexical Analysis (scanner) Phase 2: Syntactic Analysis (parser) Phase 3: Parse Tree

# Submission Details:

* Deadline: Saturday 27/5/2023 11:59 PM
* Format:
  + You should submit your project that contains two grammar files: JJ and JJT files + report + video recording.
  + Your project should be submitted via blackboard by the **group Leader**
  + You have only **two attempts** for submission.
* Discussion: Group and individual discussions will be scheduled later.
* Important note:
  + 20% will be deducted from the project score for every day the project is late after the due date. No projects will be accepted once they are three or more days late.
  + Any form of **cheating** or **plagiarism** will not be accepted under any circumstances, and the group will get ZERO out of 20.

**Phase 1:**

***[Be Creative when you design your language and do not copy C or Java rules]***

***Part 1: Regular Expressions of the Tokens***

* Identify the basic tokens of your language in Table 1: 
  + arithmetic operations example: addition, subtraction, power … etc (at least 5)
  + relational operations: equality, inequality, less than…etc (at least 5)
  + logical operators: AND, OR, NOT… etc (at least 3)
  + punctuation Marks: colon, semi-colon,….etc (as you need in your language)
  + Identifiers
  + Digits (integer and float)
  + White Spaces: one or more spaces, Tab and new line (it should be discarded)
  + Keywords
  + Comments
  + You can add other tokens as you need.

***Note: Revise Lab 1***

|  |  |
| --- | --- |
| TOKEN NAME | Regular Expression |
|  |  |
|  |  |

Table

***Part 2: Statements***

* Your language should have these statements:
  + Arithmetic statements
  + Relational statements
  + Logical statements
  + Boolean statements
  + Conditional statements: if-then, if-then-else
  + One Iterative statement such as for or while.
* Also, your language should contain the following:
  + Variable Declaration: String datatype, Integer datatype and another datatype from your choice.
  + Constant variable declaration.
  + Two data structures of your choice such as list, array.
* Provide an example for each statement in the following table:

|  |  |
| --- | --- |
| Statement Type | Code |
|  | *Note:* (*write jCC Code of lexical specification part)* |
|  |  |

Table

**Phase 2:**

* Write the grammar for your language parser using BNF notation in a word file.
  + Make sure that your Grammar free of problems such as left recursion and choice conflicts.
* Convert the grammar into code using JavaCC (in jj file).
* The jj code should display “syntactically correct statement” for the correct input statements. Otherwise, it displays an error message.
* Add some lexical actions to the definition of some tokens or syntactic actions to the grammar part (at least any 5 actions).

**Phase 3:**

* Use JJTree to build your grammar parse tree: (you will convert your jj file into jjt file).
* The jjt file should display the tree for an entered statement and display the message “correct statement”.

------------------------------------------------------------------------------

**Final Report Contents:**

* Cover page
* Table of teamwork: team members + responsibility
* Introduction to describe the main purpose of your language
* Table 1 of your language tokens.
* Table 2 of your language statements.
* BNF grammar **(with comments)**
* Screen shots examples of:
  + jj file **(example for each statement in Table 2)**
  + Lexical or syntactic actions
  + jjt file **(example for each statement in Table 2)**
* Appendix
  + The code of: jj grammar and jjt grammar **(with comments)**
* Record the screen when you run **only** **two** examples using jj file and **two** examples using jjt file.

--------------------------------------------------------------------------------------------------