# CS335 : ASSIGNMENT 0

### (Compiler Grammar Specification)

## Section 1 : Group Details

Kunal Chaturvedi : 14342 ( [kunalch@iitk.ac.in](mailto:kunalch@iitk.ac.in) )

Gourav Katha : 14254 ([kathag@iitk.ac.in](mailto:kathag@iitk.ac.in))

## Section 2 : T diagram



## Section 3 : BNF syntax

Please refer to the attached g4 file for the full grammar specifications.

## Section 4 : Deletions

Swift is a very vast language used for major development projects used by apple. Hence, it employs several techniques which are not useful for general usage and serve the purpose of making development easier for the developers. We have deleted several syntactic rules from the grammar for the project. Please refer the attached g4 file for the rules which are commented.

Some of them are mentioned below.

Statements : labeled\_statement | defer statement | do statement | compiler-control statement

Loop : for-in

Branch statements : guard statement

Control-transfer-statement : fallthrough | throw

Declaration : type-alias declaration | enum declaration | class declaration | protocol declaration | initializer declaration | deinitilizer declaration | extension declaration | subscript declaration | precedence-group declaration

Patterns : wildcard pattern | tuple pattern | enum-case pattern | optional pattern | “is” type pattern | pattern “as” type

Primary expression : self expression | superclass expression | closure expression | tuple expression | implicit member expression | wildcard expression | selector expression | key\_path expression

Generics, attributes, try expression, inheritance have been removed.

## Section 5 : changes / additions

Swift is a very vast language and has most of the required constructs already included in the grammar.

We allow the use of structs, arrays, loop constructs, branching, functional constructs etc.

Swift also allows for optional use of (;) at the end of statements. We will implement that.

Swift uses precedence groups instead of direct precedence. But we will use direct precedence in our project. Swift also classifies operators as binary or unary on the basis of the formatting of operators (whitespace or no whitespace eg : a+b , a + b, a+ b , a +b etc). This is included in the grammar but may be excluded as we feel that it makes writing simple expressions very strict. Function overloading is supported in swift by us.

## Section 6 : software

We use PLY for implementing our compiler in python.