*LITSPEAK* REPORT

ASSIGNMENT 5

60-214

NIKOLA ZJALIC

SELINA GABRIELE

SAMAN BUTTFUCK ARIF

­­­­­­­­­­­­­­­­­

UNIVERSITY OF WINDSOR

MARCH 30TH 2016

**LitSpeak Report**

**Purpose**

LitSpeak is a simple context-free programming language similar to Tiny or Java. This program is written in *Lit* syntax that takes influence directly from urbandictionary.com . The purpose of the programming language is to combine computer science knowledge while holding appeal and humor for the youth of today using current slang . Not only will beginners be able to learn simple programming concepts, they will also find humour in the program definitions which will ultimately grasp their attention and make them have the desire to learn more complex programming languages.

**Compiler:**

The compiler will work by identifying tokens defined in our EBNF grammar. It will take in the shortest token definition so as not to confuse *flame(program/method names)* and other identifiers. Once the tokens are taken in, it will expand them into their definitions until the terminal definitions cannot be broken down any further. The compiler does this by creating a symbol table and assigning pointers for parsing purposes.

**Ambiguity:**

One ambiguity currently exists in our *check-track* non-terminal definition, more commonly known as the *if-statement* in programming languages.

The construct for our *check-track* :

CHECK ≈ IF

REK ≈ ELSEIF

YOSEWF ≈ ELSE

SON ≈ ENDIF

It must begin with CHECK and end with SON with options for multiple REK statements optional one YOSEWF statement.

CHECK and REK must be followed by a *lit-beat (condition)* and a *track (code)*.

YOSEWF does not need a *lit-beat* but it DOES need a *track*.

The ambiguity arises when you create a parse tree for something such as

CHECK *lit-beat*

*code*

REK *lit-beat*

*code*

YOSEWF

*Code*

SON

There are multiple parse trees that can be created for such and while constructing the block of code, there is no necessity or structure map that ends the statement properly.

This will be solved by next week by creating more non-terminals, breaking the block of code down and possibly adding a recursive grammar definition.