Hai Nguyen

CPSC 323

Professor Anthony Le

September 26, 2019

**Assignment 2 (Syntax Analyzer) Documentation**

Operating System: Windows 10

Language: C++ (Visual Studio 2017)

Description:

**1. Problem Statement:**

Build a syntax analyzer using a top down parser (predictive RDP).

**2. How to use the program:**

- User enters a source code file or a single line of code.

- The program generates a stream of tokens and analyze the syntax.

- The program prints the tokens and production rules to the screen and “parse\_result.txt”  
 file.

- If the code is syntactically correct, the program will print “Finished.” at the end.  
 Otherwise, it will print an error message.

**3. Design of the program**:

int\_num, real\_num : an int or real number

id : a valid identifier

int, float, bool: keywords

**Production rules:**

<Statement> -> <Expression statement> | <Assign> | <Declare> | <If> | <While>

Assign statement: (e.g. “a = b + c;”)

<Assign> -> id = <Expression>;

<Expression> -> <Term> <ExpressionPrime>

<ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrime> | <Empty>

<Term> -> <Factor> <TermPrime>

<TermPrime> -> \* <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>

<Factor> -> - <Primary> | <Primary>

<Primary> -> id | int\_num | ( <Expression> ) | real\_num | true | false

<Empty> -> epsilon

First(<Assign>) = { id}

Expression statement: (e.g. “1 +a;”, “a;”)

<Expression Statement> -> <Expression>;

<Expression> -> <Term> <ExpressionPrime>

<ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrime> | <Empty>

<Term> -> <Factor> <TermPrime>

<TermPrime> -> \* <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>

<Factor> -> - <Primary> | <Primary>

<Primary> -> id | int\_num | ( <Expression> ) | real\_num | true | false

<Empty> -> epsilon

First(<Expression Statement>) = {-, id, int\_num, real\_num, (, true, false }

Declare statement: (e.g. “int a;”)

<Declare> -> <Type> id <MoreIDs>;

<Type> -> int | float | bool

<MoreIDs> -> , id <MoreIDs> | <Empty>

First(<Declare>) = {int, float, bool}

If statement:

<If> -> if <Conditional> then <Statement> endif

<Conditional> -> <Expression> <ConditionalPrime>

<ConditionalPrime> -> <Relop> <Expression> | <Empty>

<Relop> -> < | <= | == | <> | >= | >

First(<If>) = {if}

**4. Any Limitation:**

None

**5. Any Shortcomings:**

None

**6. Test cases:**