**COVER PAGE**

**CS323 Programming Assignments**

1. Names [ 1. Vanessa Delfin ], (4pm class [ ] or 5:30pm class [ x ] )

[ 2. Navie Vurdien ], (4pm class [ ] or 5:30pm class [ x ] )

2. Assignment Number [ 2 ]

3. Due Dates **Softcopy**  [ 04/09/2018 ], **Hardcopy** [ 04/10/2018 ]

4. Turn-In Dates **Softcopy** [ 04/09/2018 ], **Hardcopy** [ 04/10/2018 ]

5. Executable FileName [assignment2python3.6.py]

6. LabRoom [CS-200]

7. Operating System [Windows]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**To be filled out by the Instructor:**

GRADE:

COMMENTS:

**CS323 Documentation Assignment 1**

**1. Problem Statement**

The main purpose of the project is to write a syntax analyzer which checks the structure of RAT18S language and constructed production rules. The tokens are still distinguish from one another with the use of the lexical analyzer. After getting the tokens, the syntax analyzer parses the token through each function using RDP approach.

**2. How to use your program**

In the CS202 lab, use the command prompt and run the following command in the correct directory:

python assignment2python3.6.py

The program will prompt you asking for an input filename that will analyze each token. After it successfully runs the Lexer and the syntax analyzer it will ask for an output filename to output the type of token and the lexeme and the production rules associated with the source code.

**3. Design of your program**

The lexer method takes an expression. Each token in the expression is checked if it is an integer, real, space, identifier, separator and operator.

We created a syntaxAnalyzer function that takes the a line of code and type of token. It loops through the line of code and gets pass to the start functions which begins analyzing the source code. We also constructed functions for expression, expression prime, body, factor, term, term prime, condition, statement, statement list, declaration list, parameter list, function definition to which each token goes through.

<Start> ::= <Function> | <Declaration List> | <Statement> | <Expression> | <Condition>

<Function Definitions> ::= <Function> <Function Definitions Prime>

<Function Definitions Prime> ::= <Function Definitions> | ɛ

<Parameter List> ::= <Parameter> <Parameter List Prime>

<Parameter List Prime> ::= <Parameter List> | ɛ

<Declaration List> := <Declaration> | <Declaration List Prime>

<Declaration List Prime> := <Declaration List>

<Statement List> ::= <Statement> <Statement List Prime>

<Statement List Prime> ::= <Statement List> | ɛ

<Expression> ::= <Term><Expression Prime>

<Expression Prime> ::= + <Term><Expression Prime> | -<Term><Expression Prime> | ɛ

<Term> ::= <Factor> <Term Prime>

<Term Prime> ::= \* <Factor> <Term Prime>| / <Factor> <Term Prime> | ɛ

**4. Any Limitation**

None

**5. Any shortcomings**

None