**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 [ 1 ]

3. Due Dates **Softcopy**  [ 03/05/2018 ], **Hardcopy** [ 03/06/2018 ]

4. Turn-In Dates **Softcopy** [ 03/03/2018 ], **Hardcopy** [ 03/06/2018 ]

5. Executable FileName [assignment1python2.7.py]

6. LabRoom [CS-202]

7. Operating System [Linux]

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

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

GRADE:

COMMENTS:

**CS323 Documentation Assignment 1**

**1. Problem Statement**

This program creates a lexical analyzer for Rat18S language. It takes each token and identifies each if it is an integer, real number, separator, operator, or a keyword.

Input: An input file containing Rat18S sample code

Output: An output file identifying the type of token each token is.1qsx

**2. How to use your program**

The program can be use for both python2.7 and python3.6 version. If you are using Python2.7 run:

python assignment1python2.7.py

If you are using Python3.6 run:

python assignment1python3.6.py

The program will prompt you asking for an input filename that will analyze each token. After it successfully runs the Lexer, it will ask for an output filename to output the token types along with its respective token into a file.

**3. Design of your program**

We used python’s Enum library to classify each lexeme with it’s name and a number.

Each token held the token itself along with the Enum value in a dictionary like format. (ie. {‘token’: “tokenName”, ‘lexeme’: ‘lexemeName’})

Each of these token dictionaries were pushed into an array for output upon completion of the Lexer function.

The lexer method takes an expression. Each token in the expression is checked if it is an integer, real, space, identifier, separator and operator. Once a token is checked, it returns an integer value and is assigned to variable col. The currentState variable is then assign a state from the stateTable. If the currentState is state.REJECT, the whitespace in currentToken is replaced with no space. If the previous state is not equal to State.SPACE and is equal to currentToken, it then checks if the currentToken is a !. If it is true, the rest of the expression is a comment, else it is not a comment and it checks if the currentToken is a keyword, identifier or a separator. The token is appended the its specific state. Token is assigned to currentToken and checkToken function is assigned to currentState. However, if the currentState is not State.REJECT, then whitespace in the currentToken is replaced and assigned back to currentToken and token is added to currentToken. Lastly, if the currentState does not equal the State.SPACE and currentToken, and the currentState does not match State.REJECT, token is appended.

**4. Any Limitation**

None

**5. Any shortcomings**

None