**CPSC 323 Assignment 1 Documentation**

Safe Gergis, Ethan Nguyen, Brian Zee

**1) Problem Statement**Implement a lexical analyzer function for the Rat25S language that uses Finite State Machines to identify tokens (identifiers, integers, and reals) and returns both token types and lexemes as output. With the main program reading source code from a file and writing the recognized tokens to an output file.

**2) How to use your program**

1. Ensure that the executable files are working properly and concurrently.
2. Open command prompt
3. Open Run main.exe with three testcases in the same directory named test1.rat25s, test2.rat25s, test3.rat25s.
4. The output will be the same filenames with .txt after them.

**3) Design of your program**

We used three unordered sets to contain keywords, operators, and separators. There is also a Token struct that defines a type and lexeme .We have three functions that define FSMs for identifiers, integers, and reals. Our lexer program loops through every character in a file stream. First it checks if the character is a comment character and assigns the local variable commenting accordingly. If commenting is true or the character is a space, we continue to the next character.

1. If the character is a separator we return a token that has the type of separator and the lexeme.
2. If it is an operator, we check the next character to see if it is a double operator, we return a token of type Operator and the appropriate lexeme depending on if it’s a double operator or not.
3. If the character is an alphabetical, we get the whole word. We check if the word is a keyword against the keyword set. If not we use the identifier FSM to determine if it is a valid and return the appropriate token.
4. If the character is a digit we get the whole number including decimal points and run both our integer FSM and our reals FSM and return the appropriate token

Once the file has no more characters, we return a token that says EOF.

**4) Regular Expressions**

Identifier FSM

[a-zA-Z][a-zA-Z0-9\_]\* or L(L|D|\_)\*

Integer FSM

[0-9]+ or D+

Reals FSM

[0-9]+\.[0-9]+ or D+.D+

**5) Limitations**

None

**6) Any shortcomings**

None