1. ***Introduction***

This document contains a breakdown/report in reference to the contents of Project 1 "Building a Scanner". The scanner was written in Python for readability and simplicity. The contributors include Nafiz Imtiaz, Chris Dihenia, and Axel Alvarez.

2. ***Data Structures***

- **tokenList** *: For any final state s, tokenList[s] is the list to hold the processed tokens concludable from the final state, s.*
- **currentToken** *: A list utilized to hold temporary alphabet characters, so that id, read, and write tokens can be differentiated and appended to the* ***tokenList****.*
    - **List-** *A list is a data structure in Python that is a mutable, or changeable, ordered sequence of elements. Each element or value that is inside of a list is called an item.*

3. ***Algorithms (In Pseudocode)***

- *What outcome do we want from the program?:*
    - We want to construct a scanner that accepts a text file as an input and outputs an array/list of tokens, which we can then display to the user.

- *Functions:*
    - **Main()**

      Name: *main*
      Input: *N/A*
      Output: *0 if Passed*
      Side Effects: *Prints the array of tokens to the display*

      Pseudocode :

```
try:
        The file will be taken as an argument since it is input in command prompt
        (sys.argv returns a list of command line arguments passed to the Python
        script)

        open program file and create pointer for scanning

        tokenlist[] = Scanner(file0pen)  //input is the text file here

        close the file
        // Print the token list
        if tokenlist == "error"
                Print tokenlist (which will be an error)
```

else

Print the tokenlist in (token , token , ... , token) format

except the file does not exist:

Print error in reading the file

except the file is not entered:

Print error in inserting file name

- **Scan(file pointer)**

Name: *scan()*
Input: *A program*
Data: *tokenList : the list to hold the processed tokens*
tokenVerifier : boolean to check the longest possible token
pointer: filepointer to point to the elements in the input String
currentToken : list to hold temporary alphabet characters
string: variable to store the processed id, read, write tokens
Output: *An Array of Tokens (Strings)*
Side Effects: *N/A*

Pseudocode:
Initialize the list to store the tokens, tokenList = [ ]

Initialize a boolean variable to check the longest possible token, tokenVerifier=False

while not EOF:
while a character exists in file:
if no duplicate token
Read next input item
tokenVerifier = False //returning back to false

//characters that are not added to the tokenlist since comments
if the pointer equals to "=" or "\n" or "|t":
continue
elif pointer equals to "/":
Read next character
if pointer equals to "/":
while not equal to "\n":
Read next character
elif pointer equals to "":*
Keep Reading next character

```
                    if pointer equals to "*":
                        Read next character
                        if pointer equals to "/":
                            break
                        else:
                            continue
//Condition for division token
                else :
                    Make the tokenVerifier True
                    Append the token "division"
                continue
            elif pointer equals to "*":
                Append the token "times"
            elif pointer equals to "(":
                Append the token "left parenthesis"
            elif pointer equals to")":
                Append the token "right parenthesis"
            elif pointer equals to "+":
                Append the token "addition"
            elif pointer equals to "-":
                Append the token "subtraction"
            elif pointer equals to ":":
                Read the next input item
                if pointer equals to "=":
                    Append the token "assign"
                else:
                    return "invalid token"
                continue
//Condition to handle decimals
            elif pointer equals to ".":
                Read the next input item
                if pointer equals to digit:
                    while pointer equals to digit:
                        Read the next input item
                        if pointer equals to digit:
                            continue
                        else:
                            tokenVerifier=True
                            Append the token "number"
```

```
                    //Condition to handle numbers
                    elif  pointer equals to digit :
                        while pointer equals to digit :
                            Read the next input item
                            if pointer equals to digit :
                                continue
                            else:
                                tokenVerifier=True
                                    Append the token "number"
                        continue
                //Condition to handle the id, read, write tokens
                    elif  pointer equals to alphabet:
                    // Temporary list to hold the alphabet characters
                        currentToken= [ ]
                        while  pointer equals to alphabet:
                            Append the characters to array
                            Read the next input item

                            if pointer equals to alphabet:
                                continue
                            else:
                                tokenVerifier=True
                                Intialize string variable with the elements of
                                List currentToken[ ]
                                string.lower( )

                                if string equals to "read":
                                    Append the token "read"
                                elif  string equals to "write":
                                    Append the token "write"
                                else:
                                    Append the token "id"
                                current Token = [ ]
                    else:
                        tokenList = "error"
            if pointer equals to "":
                    break
        return the tokenList
Call the main( ) function
```

4. ***Test Cases***

Test Case 1: (Text File Content Abiding by DFA Token Rules)

```
C:\Users\cdihe\Desktop\ScannerProgram>python Scanner.py Scanner.txt
(read, times, id, number, division, times, addition, subtraction, assign, number, write)
```

```
read

/* foo
      bar */
*
five 5
/ * + - :=  .01

write
```

← **Scanner.txt (*NOT INCLUDED IN ZIP FILE)(USED FOR SELF TESTING*)**

Test Case 2: (Text File Content Not Abiding by DFA Token Rules (File Contains Any Non-valid Token in the Input File)

```
C:\Users\crysb\OneDrive\Desktop\ScannerProgram\ScannerProgram>python Scanner2.py Scanner.txt
error.
```

```
read

/* foo
      bar */
*
five 5

@
```

← **Scanner.txt (*NOT INCLUDED IN ZIP FILE)(USED FOR SELF TESTING*)**

Test Case 3: (Text File Content Abiding by DFA Token Rules (No Text File Name in Command Line Call)

```
C:\Users\crysb\OneDrive\Desktop\ScannerProgram\ScannerProgram>python Scanner2.py
Error: File Name Not Inserted : Please Insert File Name
```

```
read

/* foo
      bar */
*
five 5
```

← **Scanner.txt (*NOT INCLUDED IN ZIP FILE)(USED FOR SELF TESTING*)**

Test Case 4: (Text File Content Abiding by DFA Token Rules (Improper Text File Name in Command Line Call)

```
C:\Users\crysb\OneDrive\Desktop\ScannerProgram\ScannerProgram>python Scanner2.py Scanner
Error: ERROR READING FILE
```

```
read

/* foo
      bar */
*
five 5
```

← **Scanner.txt (*NOT INCLUDED IN ZIP FILE)(USED FOR SELF TESTING*)**

Test Case For All Included Text Files (Text File Content Abiding by DFA Token Rules (Output of Test Cases Included In Zip File)

```
C:\Users\cdihe\Desktop\ScannerProgram2>python Scanner.py test1.txt
(read, write, times, id, number)

C:\Users\cdihe\Desktop\ScannerProgram2>python Scanner.py test2.txt
(read, write, times, id, number, division, times, addition, subtraction, assign, number, write)

C:\Users\cdihe\Desktop\ScannerProgram2>python Scanner.py test3.txt
(id, left parenthesis, number, right parethesis, assign, read)

C:\Users\cdihe\Desktop\ScannerProgram2>python Scanner.py test4.txt
(id, id, assign, number, addition, number, subtraction, number, id, write)

C:\Users\cdihe\Desktop\ScannerProgram2>python Scanner.py test5.txt
(read, id, assign, number, division, number, write)

C:\Users\cdihe\Desktop\ScannerProgram2>python Scanner.py test6.txt
error.
```

5. *Acknowledgements*

Special thanks to Yuanlin Zhang (Dr. Zhang) for his useful vocal discussions regarding the contents of our project, as well as providing us with constructive advice.