# Lexical Analyzer Report

- **Names**
  - Erin Sorbella
  - Alex Chen
  - Steven Bristovitski
  - Michael Rourke
- **Date**
  - 02/01/2024
- **Course Number and details**
  - CS 361
- **Development environment/Languages**
  - The lexical analyzer was developed using Java in VSCode and uses the example.c and example2.c as the source files.
- **Files used:**
  - Example.c: This input takes an int a as input by a user and iterates through if statements whether the input number is even or odd.
  - Example2.c: This input takes two int numbers from a user as input, sums the two inputs into sum and prints it.
- **Description of the code Analyzer.Java**
  - *Package Imports*
    - Java.IO: Used here to process inputs and produce outputs
    - Java.util. *and Java.util.regex* :Are used here to import the common utilities used in Java such as ArrayLists, Data Structures and LinkedLists.
  - *Variables*
    - ArrayList < Token > tokenList: a list that holds all the tokens found in the source file.
  - *String Expressions*
    - defines the type of tokens
  - *Token Class Methods*
    - Type and value are declared for token types and values to be accepted in methods as input
    - Token Methods
      - Declares type and value properties
      - Returns both value and type
      - Uses toString method to print type and value easily.

    - *Tokenizer ArrayList method*
        - Accepts string input that will be used to for comparison with regular expressions.
        - combinedPattern combines all regular expressions into single pattern
        - Pattern and Matcher used to identify the string input and then matches it into a specific token type with the use of a while loop.
        - Adds each token into tokenList
    - *Main*
        - BufferReader method
            - Used to read the text from character input stream and buffer the characters that are read from the specified fileName (In this case: example.c)
        - Catch (IOException e)
            - used here during the input and output process when reading the file name. If the file is not found, the code will throw an exception that the file is not found.
        - For loop that prints all the found tokens within the source file
- **Description of the code example.c**
    - *Package imports*
        - stdio.h
            - import for accepting input and output
        - stdlib.h
            - general library for C to import memory allocation, conversions, and process control
    - *Main*
        - Variables
            - int a is the declared variable that will be used as input
        - The program scans for users int input and checks if the number is even or odd using a if loop
- **Description of the code example2.c**
    - *Main*
        - declares 3 ints that are 2 numbers and a sum
        - Asks the user to input 2 numbers
        - Sums the 2 numbers and prints the sum

# Analyzer.java

```
package cs361.Analyzer;

import java.util.*;
```

```
import java.util.regex.*;

import java.io.*;



public class Analyzer {

//maintain single list of tokens for a given program.

public static ArrayList<Token> tokenList = new ArrayList<>();



//regular expressions for each type of token.

public static final String keywords =
"\\b(if|else|int|double|float|struct|char|long|for|while|return|switch|case)
\\b";

public static final String identifiers = "[a-zA-Z_][a-zA-Z0-9_]*";

public static final String operators = "\\+|-|\\*|/|%|=|==|!=|<|>|
<=|>=|&&|\\|\\|";

public static final String punctuation = "[{}()\\[\\],;.]";

public static final String constants = "\\b(\\d+\\.\\d+|\\d+\\.\\d*[eE][+-]?
\\d+|\\d+[eE][+-]?\\d+|\\d+|'[^']*'|\"[^\"]*\")\\b";



//create Token class for each token found in

//given program. Can return its type and value

//as well as a toString() method.

public static class Token{

private String type;

private String value;



public Token(String type, String value)

{

this.type = type;
```

```
this.value = value;
}

public String getType()
{
return this.type;
}

public String getValue()
{
return this.value;
}

@Override
public String toString() {
return "TYPE: " + this.type + " VALUE: " + this.value + "\n";
}
}
//tokenize the program line-by-line.
//each token is identified by comparing against each regular expression.
public static ArrayList<Token> Tokenizer(String input)
{
String combinedPattern = String.join("|", keywords, identifiers, operators,
punctuation, constants);
Pattern pattern = Pattern.compile(combinedPattern);
Matcher matcher = pattern.matcher(input);
```

```
while(matcher.find())
{
String tokenType = "";
String tokenValue = matcher.group();
if (tokenValue.matches(keywords)) {
tokenType = "Keyword";
} else if (tokenValue.matches(identifiers)) {
tokenType = "Identifier";
} else if (tokenValue.matches(operators)) {
tokenType = "Operator";
} else if (tokenValue.matches(punctuation)) {
tokenType = "Punctuation";
} else if (tokenValue.matches(constants)){
tokenType = "Constant";
}
//add each token to overall token list
tokenList.add(new Token(tokenType, tokenValue));
}

return tokenList;
}

//Main function. Reads file "example.c" and parses each line
//through Tokenizer.
public static void main(String[] args)
{
```

```
BufferedReader reader;

try {

String line;

reader = new BufferedReader(new FileReader("example2.c"));

while ((line = reader.readLine()) != null)

{

Tokenizer(line);

}

reader.close();

}

catch (IOException e) {

System.out.println("File not found.");

}

for (Token token: tokenList)

{

System.out.print(token);

}

}

}
```

## Example.c

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int a;
    printf("Enter whole number:\n");
    scanf("%d", &a);

    if (a % 2 == 0)
        printf("c is even.\n");
    else
```

```
        printf("c is odd.\n");
    return 0;
}
```

## Example2.c

```c
#include <stdio.h>

int main() {
    //Find sum of two numbers
    int num1, num2, sum;

    printf("Enter the first integer: ");
    scanf("%d", &num1);

    printf("Enter the second integer: ");
    scanf("%d", &num2);

    sum = num1 + num2;

    printf("The sum of %d and %d is %d\n", num1, num2, sum);

    return 0;
}
```