Comp 2140 Lab Assignment 1: Regular expression in Java (8 points)

1 Due Date and Submission

In your lab section during the week September 19/26. To be submitted 15 minutes before the end of your lab section. You can also submit in the labs one week earlier. This assignment (and all the other assignments in this course) is an individual assignment and should be done independently. Make sure that you follow Senate Bylaws 31 (click here for the document). You must submit your code to our marking web site at our submission site. You will see the feedback and your mark immediately after your submission.

2 Objectives

Get familiar with regular expression. Learn to read grammar of **Tiny** language. Understand the wide application of regular expressions.

3 Assignment specification

Your task is to pick-up `identifiers` in programs written in our **Tiny** language. Click on the link (the underlined) for the grammar of the **Tiny** language.

Identifier

An identifier consists of a letter followed by any number of letters or digits. The following are examples of identifiers:

- `x, x2, xx2, x2x`,
- `End, END2`.

Note that `End` is an identifier while `END` is a keyword in the **Tiny** language. The following are not identifiers:

- `IF, WRITE, READ`, ... (keywords are not counted as identifiers)
- `2x` (identifier can not start with a digit)
- Strings in comments are not identifiers.

You will write a method that pick out the identifiers from a text file. For example, given a sample input:

```
INT f2(INT x, INT y )
BEGIN
   z := x*x - y*y;
   RETURN z;
END
INT MAIN F1()
BEGIN
   INT x;
   READ(x, "A41.input");
   INT y;
   READ(y, "A42.input");
   INT z;
   z := f2(x,y) + f2(y,x);
   WRITE (z, "A4.output");
END
```

Your code should return a set of identifiers that consists of {`f2, x, y, z, F1`}. Please note that in this sample input program, the following are not counted as identifiers:

- `A41, input, output`: they are quoted hence they are not treated as identifiers;
- `INT, READ` etc.: They are keywords used in our **Tiny** language hence they should not be picked up.

Here are a few test cases for the assignment: case 1, case 2, case 3, case 4, case 5, case 6. For those cases, their corresponding ID counts are 5, 4, 6, 7, 8, 9, respectively. To make your task simpler, In this assignment you can suppose that there are no comments in the input program.

You will write two different programs to do this as specified below.

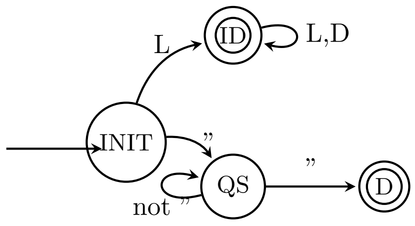
3.1 A11: coding from scratch

The first approach is the accomplish the task from scratch without using any tools. This approach also motivates the introduction of DFA in Assignment 2. Program A11.java is not supposed to use regular expressions, not regex package, not any methods involving regular expression in String class or other classes. Your program should use the most primitive method, i.e. look at characters one by one, and write a loop to check whether they are quoted strings, identifiers, etc.

A simplified version of the algorithm can be depicted by Algorithm 1. It gets a set of *identifiers* from an input string x. The algorithm starts with the initial (”INIT”) state, and scans the characters one by one. Once it sees a letter, it goes to the ”ID” state. In the ”ID” state, it expects to see more letter or digits, until it sees a character other than letter or digit. At this point, it exits the ”ID” states, and goes back to the initial state ”INIT”. The algorithm needs to be expanded to deal with quoted strings and keywords. For quoted strings, you can remove them first before you pick the identifiers. For keywords, you can check whether a token belongs to the keyword set before adding into the *identifiers* set.

Algorithm 1: The algorithm for obtaining identifiers from an input string.

```
Input: An input string $x$.
Output: a set of identifiers in $x$
1  $state$=”INIT”;
2  $token$=”” ;
3  $identifiers$={} ;
4  while $(c=nextChar())!=end\_of\_string\_x$ do
5      if $c$ $isLetter$ then
6          $state$=”ID”;
7          $token$ =$token$+c;
8      if $state$ $is$ ”ID” then
9          if $c$ $is$ $letter$ $or$ $digit$ then
10             $state$=”ID”;
11             $token$=$token$+c;
12         else
13             add $token$ to $identifiers$;
14             token=””;
15             $state$=”INIT”;
```



We provide the starter code for A11 as follows. You need to expand it to deal with quoted strings and keywords.

```java
import java.io.FileReader;
import java.io.BufferedReader;
import java.util.Set;
import java.util.HashSet;

public class A11 {
  //check whether the char is a letter
  static boolean isLetter(int character) {
    return (character >= 'a' && character <= 'z') || (character >= 'A' && character <= 'Z');
  }

  // check whether the char is a letter or digit
  static boolean isLetterOrDigit(int character) {
    return isLetter(character) || (character >= '0' && character <= '9');
  }

  public static Set<String> getIdentifiers(String filename) throws Exception{

                String[] keywordsArray = { "IF", "WRITE", "READ", "RETURN", "BEGIN",
                                "END", "MAIN", "INT", "REAL" };
```

```
        Set<String> keywords = new HashSet();
        Set<String> identifiers = new HashSet();
        for (String s : keywordsArray) {;
                keywords.add(s);
        }
        String state="INIT"; // Initially it is in the INIT state.

        StringBuilder code = new StringBuilder();
        BufferedReader br = new BufferedReader(new FileReader(filename));
        String line;
        while ((line = br.readLine()) != null) {
                code=code.append(line+"\n");
        } // read the text line by line.
        code =code.append('$'); //add a special symbol to indicate the end of file.

        int len=code.length();
        String token="";
        for (int i=0; i<len; i++) { //look at the characters one by one
                char next_char=code.charAt(i);

                if (state.contentEquals("INIT")){
                    if (isLetter(next_char)){
                        state="ID";  // go to the ID state
                        token=token+next_char;
                    } //ignore everything if it is not a letter

                }else if (state.equals("ID")) {
                        if (isLetterOrDigit(next_char)) { //take letter or digit if it is
                            in ID state
                          token=token+next_char;
                        } else { // end of ID state
                                identifiers.add(token);
                                token="";
                                state="INIT";
                        }

                }

        }

        return identifiers;

  public static void main(String[] args) throws Exception{
    Set<String> ids=getIdentifiers("A1.tiny");
    for (String id:ids) System.out.println(id);
  }
}
```

How to Run Java in Command Line

You will have much more control if you run java in command line instead of an IDE. To compile your java program in command line, you type

```
javac A11.java
```

You may see a warning message like `Note: Recompile with -Xlint:unchecked for details`. You can simply ignore it. Then, you can run your program by typing:

```
java A11
```

There may be a class not found error. In this case, check whether you have a package declaration. If there is one, remove the package declaration. If class not found error is still there, try the following command:

```
java -classpath . A11
```

`-classpath .` tells the JVM to look for the class under the current directory. The dot (.) here means current directory.

3.2 A12

Program A12.java will use regular expression in java.util.regex. Here is a tutorial for Java regex. There are many ways to solve the problem. One approach is described in the starter code below– it read the code line by line. In each line it find quoted strings and replace them with empty string. Then find identifiers and put them into a set if they are not keywords.

```java
import java.io.*;
import java.util.HashSet;
import java.util.Set;
import java.util.regex.*;
public class A12 {
        public static Set<String> getIdRegex(String filename) throws Exception{
                String[] keywordsArray = { "IF", "WRITE", "READ", "RETURN", "BEGIN","END", "MAIN",
                    "INT", "REAL" };
                Set<String> keywords = new HashSet();
                Set<String> identifiers = new HashSet();
                for (String s : keywordsArray)
                        keywords.add(s);

                FileReader reader = new FileReader(filename);
                BufferedReader br = new BufferedReader(reader);
                String line;
                //Pattern idPattern = ......;
                //Pattern quotedStringPattern = .....;
                while ((line = br.readLine()) != null) {
                        Matcher m_quotedString = quotedStringPattern.matcher(line);
                        String lineWithoutQuotedStrings = m_quotedString.replaceAll("");
                        Matcher m = idPattern.matcher(lineWithoutQuotedStrings);
                        while (m.find()) {
                                        String id = line.substring(m.start(), m.end());
                                        if (!keywords.contains(id))
                                                identifiers.add(id);
                        }
                }
                return identifiers;
     }
     public static void main(String[] args) throws Exception{
        Set<String> ids=getIdRegex("A1.tiny");
        for (String id :ids)
                System.out.println(id);
     }
}
```

4 Marking Scheme

```
yourMark=0;
if (A11.java, A12.java are not sent properly) return;
for (each of A11, A12)
     if (it is compiled correctly)  yourMark+=1;
for (each of A11, A12){
         if (your java program reads A1.tiny && generates corrent results)
                    for (each of the 6 tests cases)
                          if (it is correct)   yourMark+=0.5;
         if (youCode.length() among the top 6 students) yourMark+=0.5;

 }
 for (each day of your late submission)  yourMark=yourMark*0.8;
```

5 Bonus Mark

If your code is among the top six in terms of length, you will receive 0.5 point for A11 or A12.

"Talk is cheap. Show me the code." –Linus Torvalds