CSE 142 Section Handout #6
Challenge Sheet

You are not expected or required to solve these problems. These problems are designed for students who want a fun, extra challenge to test their skills on harder programming problems. Have fun!

By this point in the quarter, hopefully you've had the occasion to use the Output Comparison Tool linked on the course website under the assignments tab (if you haven't, check it out!). It compares expected output to the actual output a program produces and prints out all the differences. For this week's challenge, you will help implement a similar tool called the Output Comparison 2L. We have written the file prompting in main, as shown below, but we still need a method to compare the files and report differences. Write a static method called `compareFiles` that takes two Scanners, one for expected output and one for actual, prints out all differences found, including the line numbers and any extra lines in one file that don't appear in the other, and returns the total number of errors found.

```
public static void main(String[] args) throws FileNotFoundException {
   Scanner console = new Scanner(System.in);
   System.out.println("Welcome to the Output Comparison 2L!");
   System.out.print("Expected output file name: ");
   Scanner expected = new Scanner(new File(console.next()));
   System.out.print("Actual output file name: ");
   Scanner actual = new Scanner(new File(console.next()));
   int errors = compareFiles(expected, actual);
   System.out.println(errors + " difference(s) found!");
}
```

expected.txt:

```
This is a test for the
Output Comparison 2L
It should show all lines that are
not matching like this one

It also shows extra lines!!
```

actual.txt:

```
This is a test for the
Output Comparison 2L
It should show all lines that are
not matching loke this one is wrong
```

For example, running main with the above files should produce the following output:

```
Welcome to the Output Comparison 2L!
Expected output file name: expected.txt
Actual output file name: actual.txt
Line 4
  expected: not matching like this one
  actual  : not matching loke this one is wrong
Line 5
  expected:
Line 6
  expected: It also shows extra lines!!
3 difference(s) found!
```

CSE 142 Section Handout #6

Solutions

```
public static int compareFiles(Scanner expected, Scanner actual) {
   int errors = 0;
   int line = 0;
   while (expected.hasNextLine() && actual.hasNextLine()) {
      line++;
      String nextExpected = expected.nextLine();
      String nextActual = actual.nextLine();
      if (!nextActual.equals(nextExpected)) {
         errors++;
         System.out.println("Line " + line);
         System.out.println("  expected: " + nextExpected);
         System.out.println("  actual  : " + nextActual);
      }
   }

   while (expected.hasNextLine()) {
      line++;
      errors++;
      String nextExpected = expected.nextLine();
      System.out.println("Line " + line);
      System.out.println("  expected: " + nextExpected);
   }

   while (actual.hasNextLine()) {
      line++;
      errors++;
      String nextActual = actual.nextLine();
      System.out.println("Line " + line);
      System.out.println("  actual  : " + nextActual);
   }

   return errors;
}
```

You might notice that this solution has some redundancy in the part that scans through the end of the files to check for additional lines appearing in one but not the other. If you would like some extra challenge, try to eliminate this redundancy by creating another static method to scan through these additional lines. You will have to decide what kind(s) of parameter(s) you will need and if you want to make a return from that method.

CSE 142 Section Handout #6

Solutions

Eliminated redundancy solution:

```
public static int compareFiles(Scanner expected, Scanner actual) {
   int errors = 0;
   int line = 0;
   while (expected.hasNextLine() && actual.hasNextLine()) {
      line++;
      String nextExpected = expected.nextLine();
      String nextActual = actual.nextLine();
      if (!nextActual.equals(nextExpected)) {
         errors++;
         System.out.println("Line " + line);
         System.out.println("  expected: " + nextExpected);
         System.out.println("  actual  : " + nextActual);
      }
   }

   int additionalErrors = checkEndOfFile(expected, "expected", line);
   line += additionalErrors;
   errors += additionalErrors;

   additionalErrors = checkEndOfFile(actual, "actual ", line);
   line += additionalErrors;
   errors += additionalErrors;

   return errors;
}

public static int checkEndOfFile(Scanner fileScan, String type, int line) {
   int errors = 0;
   while (fileScan.hasNextLine()) {
      errors++;
      String nextLine = fileScan.nextLine();
      System.out.println("Line " + (errors + line));
      System.out.println("  " + type + ": " + nextLine);
   }
   return errors;
}
```