CS 122 – Homework #7 – due Monday, April 15, 2019

{ } { } Matching Curly Braces { } { }

The purpose of this assignment is to practice with stacks. A stack is a linear data structure similar to an ArrayList, but with access limited only at one end (the top) – so the typical operations are "push" for adding something to the top of a stack and "pop" for removing the top element. Fortunately, the Java API has a built-in Stack class, so it won't be necessary for you to define what a stack is.

In this assignment, you will be matching corresponding curly braces that appear in a Java source program provided as input. First, ask the user for the name of the source file. Open the file, and then scan it for curly braces. We need to match each '{' with a '}'. Because curly braces can nest, we have a last-in-first-out organization, so the best way to model this is by using a stack.

The output from your program will be pairs of line numbers. Each pair of line numbers will indicate the location of a particular pair of curly braces that match. You should print these pairs in the order in which they are resolved. In other words, each time you find a '}', find the '{' that matches it, and output the line numbers for both. Print one pair of numbers per line of output, separated by a space.

Your program should ignore curly braces that occur inside comments and quotes. Your program will need to carefully inspect each input character to determine if it is currently inside a comment or a character constant or string literal. Also beware of escape characters, which are characters preceded by a backslash.

Finally, your program should detect if there is an imbalance of curly braces. For example, if you encounter a '}' for which there is no matching '{', output an error message and halt. Your error message should indicate the line number of the '}' that could not be matched. Also, at the end of input, if there are still '{' that never got matched, print an error message listing the line numbers of those unmatched '{' symbols.

You may assume that the input Java program will be syntactically correct, except for the possibility of missing or extra curly braces.

Example I/O:

```
Please enter name of source file:  Happy.java
The matching curly braces occur on these lines:
2 3
4 6
1 7
```

In this example, assume that Happy.java looks like this:

```
public class Happy {
  public Happy() {
  }
  public String toString() {
    return ":)";
  }
}
```