Programming Assignment 9-1

Use your class `MyStringLinkedList` to implement your own stack, called `MyStringStack`. Do this by storing an instance of `MyStringLinkedList` in your stack class, instantiating it when the constructor of `MyStringStack` is called. Then implement the 3 basic stack operations, `pop, peek, push`, in your stack class by making appropriate calls to your encapsulated linked list.

Test your stack in a main method with the following code:

```
MyStringStack stack = new MyStringStack();
stack.push("Bob");
stack.push("Harry");
stack.push("Alice");
System.out.println("Popping…"+stack.pop());
System.out.println("Peeking…."+stack.peek());
System.out.println("Popping…"+stack.pop());
```

Programming Assignment 9-2

Create a class `SymbolBalancer` that has a constructor

```
SymbolBalancer(String filename)
```

which accepts the name of a file to examine, and that also has a method

```
boolean symbolsBalanced(String delimiters)
```

The delimiters argument is a list of all pairs of delimiters that will be used by your symbol balanced method. For example, here is a possible value of the delimiters parameter:

```
"[](){}"
```

The `String` that is passed into this argument must be parsed. You can do this in a loop with repeated calls to `charAt`.

Also, you should provide an instance variable `text` that will store the text to be parsed (which you will extract from the input file).

Your method `symbolsBalanced` should return true if the open/closed pairs of delimiters specified in the delimiters argument, as they occur in the text that is being examined, are balanced; false, otherwise.

We will explain more formally how to read a file in a later lesson. For now, use the `readFile()` method provided below to store the file content in a String named text. The following sample code shows how this can be done:

```
void readFile() {

      try (BufferedReader br =
         new BufferedReader(new FileReader(filename))) {
            String currentLine;
            StringBuilder builder = new StringBuilder();

            while ((currentLine = br.readLine()) != null) {
                  builder.append(currentLine);
            }

            text = builder.toString();

      } catch (IOException e) {
            e.printStackTrace();
      }
}
```

Create a `readFile()` method like the sample above. (Your constructor should call this method immediately after setting the input file name.) Test your symbol-balanced-checking code in a `main` method by reading in the `Employee.java` class (provided in a folder in this directory) using `readFile()`. (See the `readme` file for instructions about where to place the input file in your directory.) In your call to `symbolsBalanced()`, pass in the following `String` of delimiter pairs: "`[]{}<>()||`". Your `main` method should simply output either "true" or "false" to the console, indicating the result of the symbol-balanced test.

Programming Assignment 9-3

Given a nonempty string S consisting of some words separated by spaces. We want to reverse every word in S.
For example, given S = “we test coders”, your function is going to return a string with every word in S reversed and separated by spaces. So the result for the above example would be “ew tset sredoc”.
Hint: You can use a stack to solve this problem.