```java
import static org.junit.Assert.assertEquals;

import org.junit.Test;

import components.set.Set;
import components.set.Set1L;

public class StringReassemblyTest {

    // routine
    @Test
    public void testcombination_apple_lead_2() {
        String str1 = "apple";
        String str2 = "lead";
        int overlap = 2;
        String combination = StringReassembly.combination(str1, str2, overlap);
        assertEquals("applead", combination);
    }

    // boundary
    @Test
    public void testcombination_empty_empty_0() {
        String str1 = "";
        String str2 = "";
        int overlap = 0;
        String combination = StringReassembly.combination(str1, str2, overlap);
        assertEquals("", combination);
    }

    // challenging
    @Test
    public void testcombination_apple_apple_5() {
        String str1 = "apple";
        String str2 = "apple";
        int overlap = 5;
        String combination = StringReassembly.combination(str1, str2, overlap);
        assertEquals("apple", combination);
    }

    // boundary
    @Test
    public void addToSetAvoidingSubstrings_empty_empty() {
        String str = "";
        Set<String> strSet = new Set1L<>();
        Set<String> strSetTrue = strSet.newInstance();
        strSetTrue.add("");
        StringReassembly.addToSetAvoidingSubstrings(strSet, str);
        assertEquals("", str);
        assertEquals(strSetTrue, strSet);
    }

    // challenging
    @Test
    public void addToSetAvoidingSubstrings_apple_set_challenging1() {
        String str = "apple";
        Set<String> strSet = new Set1L<>();
        strSet.add("tree");
```

```
58          strSet.add("applesauce");
59
60          Set<String> strSetTrue = new Set1L<>();
61          strSetTrue.add("tree");
62          strSetTrue.add("applesauce");
63          StringReassembly.addToSetAvoidingSubstrings(strSet, str);
64          assertEquals("apple", str);
65          assertEquals(strSetTrue, strSet);
66      }
67
68      // challenging 2
69      @Test
70      public void addToSetAvoidingSubstrings_apple_set_challenging2() {
71          String str = "apple";
72          Set<String> strSet = new Set1L<>();
73          Set<String> strSetTrue = new Set1L<>();
74          strSet.add("tree");
75          strSet.add("app");
76          strSetTrue.add("tree");
77          strSetTrue.add("apple");
78          StringReassembly.addToSetAvoidingSubstrings(strSet, str);
79          assertEquals("apple", str);
80          assertEquals(strSetTrue, strSet);
81      }
82
83      // routine
84      @Test
85      public void addToSetAvoidingSubstrings_apple_set() {
86          String str = "apple";
87          Set<String> strSet = new Set1L<>();
88          strSet.add("tree");
89          strSet.add("cattle");
90          strSet.add("cherry");
91
92          Set<String> strSetTrue = new Set1L<>();
93          strSetTrue.add("tree");
94          strSetTrue.add("cattle");
95          strSetTrue.add("cherry");
96          strSetTrue.add("apple");
97          StringReassembly.addToSetAvoidingSubstrings(strSet, str);
98          assertEquals("apple", str);
99          assertEquals(strSetTrue, strSet);
100     }
101
102 }
103
```