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

import org.junit.Test;

import components.set.Set;
import components.set.Set2;
import components.simplereader.SimpleReader;
import components.simplereader.SimpleReader1L;
import components.simplewriter.SimpleWriter;
import components.simplewriter.SimpleWriter1L;

public class StringReassemblyTest {

    /*
     * Tests for combination
     */

    // Testing prefix (s1 as substring) (routine case)
    @Test
    public void testForCombination1() {
        String s1 = "Comp";
        String s2 = "mputers";
        int overlap = StringReassembly.overlap(s1, s2);
        String s3 = StringReassembly.combination(s1, s2, overlap);
        String expectedS3 = "Computers";
        assertEquals(s3, expectedS3);
    }

    // Testing long strings (challenging case)
    @Test
    public void testForCombination2() {
        String s1 = "We have lectu";
        String s2 = "lecture today!";
        int overlap = StringReassembly.overlap(s1, s2);
        String s3 = StringReassembly.combination(s1, s2, overlap);
        String expectedS3 = "We have lecture today!";
        assertEquals(s3, expectedS3);
    }

    // Testing with numbers and symbols (challenging case)
    @Test
    public void testForCombination3() {
        String s1 = "3/30/2";
        String s2 = "0/2023";
        int overlap = StringReassembly.overlap(s1, s2);
        String s3 = StringReassembly.combination(s1, s2, overlap);
        String expectedS3 = "3/30/2023";
        assertEquals(s3, expectedS3);
    }

    /*
     * Tests for addToSetAvoidingSubstrings
     */

    // Testing routine case where s1 is substring
```

```java
@Test
public void testForAddToSetAvoidingSubstrings1() {
    Set<String> set = new Set2<>();
    set.add("hello");
    set.add("there");
    String s1 = "llo";
    StringReassembly.addToSetAvoidingSubstrings(set, s1);
    Set<String> expectedSet = new Set2<>();
    expectedSet.add("hello");
    expectedSet.add("there");
    assertEquals(expectedSet, set);
}

// Testing routine case where s1 is NOT substring
@Test
public void testForAddToSetAvoidingSubstrings2() {
    Set<String> set = new Set2<>();
    set.add("quick");
    set.add("brown");
    String s1 = "fox!";
    StringReassembly.addToSetAvoidingSubstrings(set, s1);
    Set<String> expectedSet = new Set2<>();
    expectedSet.add("quick");
    expectedSet.add("brown");
    expectedSet.add("fox!");
    assertEquals(expectedSet, set);
}

// Testing challenging case with letters, numbers, and symbols
@Test
public void testForAddToSetAvoidingSubstrings3() {
    Set<String> set = new Set2<>();
    set.add("jonny#!@1245!");
    String s1 = "y#!@12";
    StringReassembly.addToSetAvoidingSubstrings(set, s1);
    Set<String> expectedSet = new Set2<>();
    expectedSet.add("jonny#!@1245!");
    assertEquals(expectedSet, set);
}

// Testing challenging case with numbers
@Test
public void testForAddToSetAvoidingSubstrings4() {
    Set<String> set = new Set2<>();
    set.add("10100110101001");
    String s1 = "110101";
    StringReassembly.addToSetAvoidingSubstrings(set, s1);
    Set<String> expectedSet = new Set2<>();
    expectedSet.add("10100110101001");
    assertEquals(expectedSet, set);
}

/*
 * Tests for linesFromInput
 */
```

```
// Routine case with just strings
@Test
public void testForLinesFromInput1() {
    SimpleReader inFile = new SimpleReader1L(
            "data/linesFromInputTestEasy.txt");
    Set<String> set = StringReassembly.linesFromInput(inFile);
    Set<String> expectedSet = new Set2<>();
    expectedSet.add("Hello, my name is John");
    expectedSet.add("I work at Kroger");
    expectedSet.add("I like to play basketball");
    assertEquals(expectedSet, set);
}

// Challenging case with numbers, strings, symbols, and empty line
@Test
public void testForLinesFromInput2() {
    SimpleReader inFile = new SimpleReader1L(
            "data/linesFromInputTestHard.txt");
    Set<String> set = StringReassembly.linesFromInput(inFile);
    Set<String> expectedSet = new Set2<>();
    expectedSet.add("11101010010110");
    expectedSet.add("jon.doe@gmail.com");
    expectedSet
            .add("Weather @2:38PM: 62 degrees, cloudy, & UV index is 4!");
    expectedSet.add("              Auf Wiedersehen!!");
    assertEquals(expectedSet, set);
}

/*
 * Tests for printLineWithSeparators
 */

// Routine case with just strings
@Test
public void testForPrineLineWithSeparators1() {
    SimpleWriter out = new SimpleWriter1L(
            "data/printLineWithSeparatorsTestEasy.txt");
    SimpleReader inFile = new SimpleReader1L(
            "data/printLineWithSeparatorsTestEasy.txt");
    String text = "Hello~my name~is Jon";
    StringReassembly.printWithLineSeparators(text, out);
    Set<String> set = StringReassembly.linesFromInput(inFile);
    Set<String> expectedSet = new Set2<>();
    expectedSet.add("Hello");
    expectedSet.add("my name");
    expectedSet.add("is Jon");
    assertEquals(expectedSet, set);
}

// Challenging case with strings, numbers, symbols, and blank line
@Test
public void testForPrineLineWithSeparators2() {
    SimpleWriter out = new SimpleWriter1L(
            "data/printLineWithSeparatorsTestHard.txt");
```

```
        SimpleReader inFile = new SimpleReader1L(
                "data/printLineWithSeparatorsTestHard.txt");
        String text = "1011010011~~2 + 2 = 4~I ran 5 mi @8:00 pace == 40 min of running.";
        StringReassembly.printWithLineSeparators(text, out);
        Set<String> set = StringReassembly.linesFromInput(inFile);
        Set<String> expectedSet = new Set2<>();
        expectedSet.add("1011010011");
        expectedSet.add("2 + 2 = 4");
        expectedSet.add("I ran 5 mi @8:00 pace == 40 min of running.");
        assertEquals(expectedSet, set);
    }
}
```