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

import org.junit.Test;

import components.set.Set;
import components.set.Set1L;
import components.simplereader.SimpleReader;
import components.simplereader.SimpleReader1L;
import components.simplewriter.SimpleWriter;
import components.simplewriter.SimpleWriter1L;

public class StringReassemblyTest {

    @Test
    public void testOverlap_stream() {
        String str1 = "broadcast";
        String str2 = "casting";
        int overlap = StringReassembly.overlap(str1, str2);
        assertEquals(4, overlap);
    }

    @Test
    public void testOverlap_prefixsuffix() {
        String str1 = "prefixsuffix";
        String str2 = "ixsuffixplus";
        int overlap = StringReassembly.overlap(str1, str2);
        assertEquals(8, overlap);
    }

    @Test
    public void testOverlap_nointersect() {
        String str1 = "nointersect";
        String str2 = "completely";
        int overlap = StringReassembly.overlap(str1, str2);
        assertEquals(0, overlap);
    }

    @Test
    public void testCombination_stream() {
        String str1 = "stre";
        String str2 = "ream";
        int overlap = 2;
        String combine = StringReassembly.combination(str1, str2, overlap);
        assertEquals("stream", combine);
    }

    @Test
    public void testCombination_prefixsuffix() {
        String str1 = "overlap";
        String str2 = "lapping";
        int overlap = 3;
        String combine = StringReassembly.combination(str1, str2, overlap);
        assertEquals("overlapping", combine);
    }

    @Test
    public void testAddToSetAvoidingSubstrings_1() {
```

```
58          Set<String> test = new Set1L<>();
59          test.add("abc");
60          test.add("def");
61          test.add("jkl");
62          String str = "jklmn";
63          Set<String> expected = new Set1L<>();
64          expected.add("abc");
65          expected.add("def");
66          expected.add("jklmn");
67          StringReassembly.addToSetAvoidingSubstrings(test, str);
68          assertEquals(expected, test);
69      }
70
71      @Test
72      public void testAddToSetAvoidingSubstrings_2() {
73          Set<String> test = new Set1L<>();
74          test.add("123");
75          test.add("456");
76          String str = "456789";
77          Set<String> expected = new Set1L<>();
78          expected.add("123");
79          expected.add("456789");
80          StringReassembly.addToSetAvoidingSubstrings(test, str);
81          assertEquals(expected, test);
82      }
83
84      @Test
85      public void testAssemble_1() {
86          Set<String> test = new Set1L<>();
87          test.add("part1 ");
88          test.add("1 part2");
89          test.add("part2 part3");
90          Set<String> expected = new Set1L<>();
91          expected.add("part1 part2 part3");
92          StringReassembly.assemble(test);
93          assertEquals(expected, test);
94      }
95
96      @Test
97      public void testAssemble_2() {
98          Set<String> test = new Set1L<>();
99          test.add("123");
100         test.add("234");
101         test.add("345");
102         test.add("456");
103         Set<String> expected = new Set1L<>();
104         expected.add("123456");
105         StringReassembly.assemble(test);
106         assertEquals(expected, test);
107     }
108
109     @Test
110     public void testPrintWithLineSeparators_1() {
111         SimpleWriter out = new SimpleWriter1L("test.txt");
112         SimpleReader in = new SimpleReader1L("test.txt");
113         String text = "ABC~DEF~GHI";
114         String expected = "ABC\nDEF\nGHI";
```

```
115         StringReassembly.printWithLineSeparators(text, out);
116         String test1 = in.nextLine();
117         String test2 = in.nextLine();
118         String test3 = in.nextLine();
119         in.close();
120         out.close();
121         assertEquals(expected, test1 + "\n" + test2 + "\n" + test3);
122     }
123 
124     @Test
125     public void testPrintWithLineSeparators_2() {
126         SimpleWriter out = new SimpleWriter1L("test.txt");
127         SimpleReader in = new SimpleReader1L("test.txt");
128         String text = "123~456~789";
129         String expected = "123\n456\n789";
130         StringReassembly.printWithLineSeparators(text, out);
131         String test1 = in.nextLine();
132         String test2 = in.nextLine();
133         String test3 = in.nextLine();
134         in.close();
135         out.close();
136         assertEquals(expected, test1 + "\n" + test2 + "\n" + test3);
137     }
138 
139     @Test
140     public void testLinesFromInput_1() {
141         SimpleReader in = new SimpleReader1L("xd.txt");
142         Set<String> test = new Set1L<>();
143         Set<String> expected = new Set1L<>();
144         expected.add("Bucks -- Beat");
145         expected.add("Go Bucks");
146         expected.add("o Bucks -- B");
147         expected.add("Beat Mich");
148         expected.add("Michigan~");
149         test = StringReassembly.linesFromInput(in);
150         assertEquals(test, expected);
151     }
152 }
153 
```