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

public class StringReassemblyTest {

    /*
     * Tests of overlap
     */
    @Test
    public void testOverlap_WAZ_ZAW() {
        String str1 = "WAZ";
        String str2 = "ZAW";
        int max = StringReassembly.overlap(str1, str2);
        assertEquals(1, max);
    }

    @Test
    public void testOverlap_QWERQWE_RQWEWQR() {
        String str1 = "QWERQWE";
        String str2 = "RQWEWQR";
        int max = StringReassembly.overlap(str1, str2);
        assertEquals(4, max);
    }

    /*
     * Tests of combination
     */
    @Test
    public void testCombination1() {
        String str1 = "AGCT";
        String str2 = "GCTQ";
        int max = 3;
        String comb = StringReassembly.combination(str1, str2, max);
        assertEquals("AGCTQ", comb);
    }

    @Test
    public void testCombination2() {
        String str1 = "QWERTYU";
        String str2 = "TYUIOPL";
        int max = 3;
        String comb = StringReassembly.combination(str1, str2, max);
        assertEquals("QWERTYUIOPL", comb);
    }

    /*
     * Tests of addToSetAvoidingSubstrings
     */
    @Test
    public void testAddToSetAvoidingSubstrings1() {
        Set<String> strSet = new Set1L<>();
        Set<String> strSetExpected = new Set1L<>();
        strSet.add("QWE");
        strSet.add("WER");
        strSet.add("TY");
        strSetExpected.add("QWE");
        strSetExpected.add("WER");
        strSetExpected.add("RTY");
        String str = "RTY";
        StringReassembly.addToSetAvoidingSubstrings(strSet, str);
        assertEquals(strSetExpected, strSet);
        assertEquals("RTY", str);
    }
```

```
 72
 73     @Test
 74     public void testAddToSetAvoidingSubstrings2() {
 75         Set<String> strSet = new Set1L<>();
 76         Set<String> strSetExpected = new Set1L<>();
 77         strSet.add("ZXC");
 78         strSet.add("VBN");
 79         strSet.add("EF");
 80         strSet.add("DE");
 81         strSetExpected.add("ZXC");
 82         strSetExpected.add("VBN");
 83         strSetExpected.add("DEF");
 84         String str = "DEF";
 85         StringReassembly.addToSetAvoidingSubstrings(strSet, str);
 86         assertEquals(strSetExpected, strSet);
 87         assertEquals("DEF", str);
 88     }
 89
 90     /*
 91      * Tests of linesFromInput
 92      */
 93     @Test
 94     public void testlinesFromInput1() {
 95         Set<String> strSet = new Set1L<>();
 96         strSet.add("QWERTRQW");
 97         strSet.add("SADAFAQW");
 98         strSet.add("TRQWERQW");
 99         SimpleReader inFile = new SimpleReader1L("testF.txt");
100         Set<String> s = StringReassembly.linesFromInput(inFile);
101         assertEquals(s, strSet);
102         inFile.close();
103     }
104
105     @Test
106     public void testlinesFromInput2() {
107         Set<String> strSet = new Set1L<>();
108         strSet.add("QWE");
109         strSet.add("SAD");
110         strSet.add("TRQ");
111         SimpleReader inFile = new SimpleReader1L("testS.txt");
112         Set<String> s = StringReassembly.linesFromInput(inFile);
113         assertEquals(s, strSet);
114         inFile.close();
115     }
116
117     /*
118      * Tests of bestOverlap
119      */
120     @Test
121     public void testBestOverlap1() {
122         Set<String> strSet = new Set1L<>();
123         strSet.add("QWERTRQW");
124         strSet.add("SADAFAQW");
125         strSet.add("TRQWERQW");
126         String[] bestTwo = new String[2];
127         int num = StringReassembly.bestOverlap(strSet, bestTwo);
128         assertEquals(4, num);
129     }
130
131     @Test
132     public void testBestOverlap2() {
133         Set<String> strSet = new Set1L<>();
```

```
134         strSet.add("QWEASDZXC");
135         strSet.add("ZXCQASDQ");
136         strSet.add("QWEASFDZXCF");
137         String[] bestTwo = new String[2];
138         int num = StringReassembly.bestOverlap(strSet, bestTwo);
139         assertEquals(3, num);
140     }
141 
142     /*
143      * Tests of assemble
144      */
145     @Test
146     public void testassemble1() {
147         Set<String> strSet = new Set1L<>();
148         Set<String> strSetExpected = new Set1L<>();
149         strSet.add("QWERTRQW");
150         strSet.add("SADAFAQW");
151         strSet.add("TRQWERQW");
152         strSetExpected.add("SADAFAQWERTRQWERQW");
153         StringReassembly.assemble(strSet);
154         assertEquals(strSetExpected, strSet);
155     }
156 
157     @Test
158     public void testassemble2() {
159         Set<String> strSet = new Set1L<>();
160         Set<String> strSetExpected = new Set1L<>();
161         strSet.add("QWEASDZXC");
162         strSet.add("ZXCQASDQ");
163         strSet.add("QWEASFDZXCF");
164         strSetExpected.add("QWEASDZXCQASDQWEASFDZXCF");
165         StringReassembly.assemble(strSet);
166         assertEquals(strSetExpected, strSet);
167     }
168 
169     /*
170      * Tests of printWithLineSeparators
171      */
172     @Test
173     public void testPrintWithLineSeparators1() {
174         SimpleWriter out = new SimpleWriter1L("testPr.txt");
175         String str = "hello~world";
176         String line1 = "hello";
177         String line2 = "world";
178         StringReassembly.printWithLineSeparators(str, out);
179         SimpleReader inFile = new SimpleReader1L("testPr.txt");
180         String test1 = inFile.nextLine();
181         String test2 = inFile.nextLine();
182 
183         assertEquals(line1, test1);
184         assertEquals(line2, test2);
185         inFile.close();
186         out.close();
187 
188     }
189 
190     @Test
191     public void testPrintWithLineSeparators2() {
192         SimpleWriter out = new SimpleWriter1L("testPr.txt");
193         String str = "hello~world~to~me";
194         String line1 = "hello";
195         String line2 = "world";
```

```
196         String line3 = "to";
197         String line4 = "me";
198         StringReassembly.printWithLineSeparators(str, out);
199         SimpleReader inFile = new SimpleReader1L("testPr.txt");
200         String test1 = inFile.nextLine();
201         String test2 = inFile.nextLine();
202         String test3 = inFile.nextLine();
203         String test4 = inFile.nextLine();
204
205         assertEquals(line1, test1);
206         assertEquals(line2, test2);
207         assertEquals(line3, test3);
208         assertEquals(line4, test4);
209         inFile.close();
210         out.close();
211     }
212
213 }
214
```