```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);
    }

    @Test
    public void testAddToSetAvoidingSubstrings2() {
        Set<String> strSet = new Set1L<>();
        Set<String> strSetExpected = new Set1L<>();
        strSet.add("ZXC");
        strSet.add("VBN");
        strSet.add("EF");
        strSet.add("DE");
        strSetExpected.add("ZXC");
        strSetExpected.add("VBN");
        strSetExpected.add("DEF");
        String str = "DEF";
        StringReassembly.addToSetAvoidingSubstrings(strSet, str);
        assertEquals(strSetExpected, strSet);
        assertEquals("DEF", str);
    }

    /*
     * Tests of linesFromInput
     */
    @Test
    public void testlinesFromInput1() {
        Set<String> strSet = new Set1L<>();
        strSet.add("QWERTRQW");
        strSet.add("SADAFAQW");
        strSet.add("TRQWERQW");
        SimpleReader inFile = new SimpleReader1L("testF.txt");
        Set<String> s = StringReassembly.linesFromInput(inFile);
        assertEquals(s, strSet);
        inFile.close();
    }

    @Test
    public void testlinesFromInput2() {
        Set<String> strSet = new Set1L<>();
        strSet.add("QWE");
        strSet.add("SAD");
        strSet.add("TRQ");
        SimpleReader inFile = new SimpleReader1L("testS.txt");
        Set<String> s = StringReassembly.linesFromInput(inFile);
        assertEquals(s, strSet);
        inFile.close();
    }

    /*
     * Tests of bestOverlap
     */
    @Test
    public void testBestOverlap1() {
        Set<String> strSet = new Set1L<>();
        strSet.add("QWERTRQW");
        strSet.add("SADAFAQW");
        strSet.add("TRQWERQW");
        String[] bestTwo = new String[2];
        int num = StringReassembly.bestOverlap(strSet, bestTwo);
        assertEquals(4, num);
    }

    @Test
    public void testBestOverlap2() {
        Set<String> strSet = new Set1L<>();
        strSet.add("QWEASDZXC");
        strSet.add("ZXCQASDQ");
        strSet.add("QWEASFDZXCF");
        String[] bestTwo = new String[2];
        int num = StringReassembly.bestOverlap(strSet, bestTwo);
        assertEquals(3, num);
    }

    /*
     * Tests of assemble
     */
    @Test
    public void testassemble1() {
        Set<String> strSet = new Set1L<>();
        Set<String> strSetExpected = new Set1L<>();
        strSet.add("QWERTRQW");
        strSet.add("SADAFAQW");
        strSet.add("TRQWERQW");
        strSetExpected.add("SADAFAQWERTRQWERQW");
        StringReassembly.assemble(strSet);
        assertEquals(strSetExpected, strSet);
    }

    @Test
    public void testassemble2() {
        Set<String> strSet = new Set1L<>();
        Set<String> strSetExpected = new Set1L<>();
        strSet.add("QWEASDZXC");
        strSet.add("ZXCQASDQ");
        strSet.add("QWEASFDZXCF");
        strSetExpected.add("QWEASDZXCQASDQWEASFDZXCF");
        StringReassembly.assemble(strSet);
        assertEquals(strSetExpected, strSet);
    }

    /*
     * Tests of printWithLineSeparators
     */
    @Test
    public void testPrintWithLineSeparators1() {
        SimpleWriter out = new SimpleWriter1L("testPr.txt");
        String str = "hello~world";
        String line1 = "hello";
        String line2 = "world";
        StringReassembly.printWithLineSeparators(str, out);
        SimpleReader inFile = new SimpleReader1L("testPr.txt");
        String test1 = inFile.nextLine();
        String test2 = inFile.nextLine();

        assertEquals(line1, test1);
        assertEquals(line2, test2);
        inFile.close();
        out.close();

    }

    @Test
    public void testPrintWithLineSeparators2() {
        SimpleWriter out = new SimpleWriter1L("testPr.txt");
        String str = "hello~world~to~me";
        String line1 = "hello";
        String line2 = "world";
        String line3 = "to";
        String line4 = "me";
        StringReassembly.printWithLineSeparators(str, out);
        SimpleReader inFile = new SimpleReader1L("testPr.txt");
        String test1 = inFile.nextLine();
        String test2 = inFile.nextLine();
        String test3 = inFile.nextLine();
        String test4 = inFile.nextLine();

        assertEquals(line1, test1);
        assertEquals(line2, test2);
        assertEquals(line3, test3);
        assertEquals(line4, test4);
        inFile.close();
        out.close();
    }

}

```