Calling with Arguments

	Name	Arguments	Actual	Expected
pass	product	2.0, 2.0	4.0	4.0
pass	product	2.0, 0.0	0.0	0.0
pass	product	2.0, 1.0	2.0	2.0
pass	product	2.0, -1.0	-2.0	-2.0
pass	product	-2.0, -1.0	2.0	2.0

Submitted files

ProductTwoNumbers.java:

```
/*****************************************************
 *                                                   *
 * PROBLEM:                                          *
 *                                                   *
 * Complete the design of the function called product  *
 * that consumes two double numbers and produces the   *
 * product of those numbers.                         *
 *                                                   *
 * see examples wrapped in check-expect.             *
 *  -with the form: @check_expect (({in}) -> (out))    *
 *                                                   *
 * @author Henrique Rebelo                           *
 *****************************************************/
public class ProductTwoNumbers {

  /**
   * Examples:
   * @check_expect (({2.0, 2.0})   -> (4.0))
   * @check_expect (({2.0, 0.0})   -> (0.0))
   * @check_expect (({2.0, 1.0})   -> (2.0))
   * @check_expect (({2.0, -1.0})  -> (-2.0))
   * @check_expect (({-2.0, -1.0}) -> (2.0))
   */
  public static double product(double x, double y) {
   return x * y;
  }
}
```

Score

5/5

2023-03-12T18:01:31Z

Calling with Arguments

	Name	Arguments	Actual	Expected
pass	product	new double[] {5}	5.0	5.0
pass	product	new double[] {5, 5}	25.0	25.0
pass	product	new double[] {-5, 5}	-25.0	-25.0
pass	product	new double[] {-5, -5}	25.0	25.0
pass	product	new double[] {3, 3, 0}	0.0	0.0

Submitted files

ProductNNumbers.java:

```
/*****************************************************
 *                                                   *
 * PROBLEM:                                          *
 *                                                   *
 * Complete the design of the function called product  *
 * that consumes n double numbers and produces the     *
 * product of those numbers.                         *
 *                                                   *
 * @required the Java "for" loop                     *
 *                                                   *
 * see examples wrapped in check-expect.             *
 *  -with the form: @check_expect (({in}) -> (out))    *
 *                                                   *
 * @author Henrique Rebelo                           *
 *****************************************************/
public class ProductNNumbers {

  /**
   * Examples:
   * @check_expect (({new double[] {5}})       -> (5.0))
   * @check_expect (({new double[] {5, 5}})    -> (25.0))
   * @check_expect (({new double[] {-5, 5}})   -> (-25.0))
   * @check_expect (({new double[] {-5, -5}})  -> (25.0))
   * @check_expect (({new double[] {3, 3, 0}}) -> (0.0))
   */
  public static double product(double [] numbers) {
    double product = 1;
      for (int i = 0; i < numbers.length; i = i + 1){
         product = numbers[i] * product;
      }
    return product;
  }
}
```

Score

5/5

2023-03-12T19:34:29Z

Calling with Arguments

	Name	Arguments	Actual	Expected
pass	product	new double[] {5}	5.0	5.0
pass	product	new double[] {5, 5}	25.0	25.0
pass	product	new double[] {-5, 5}	-25.0	-25.0
pass	product	new double[] {-5, -5}	25.0	25.0
pass	product	new double[] {3, 3, 0}	0.0	0.0

Submitted files

ProductNNumbers2.java:

```
/*****************************************************
 *                                                   *
 * PROBLEM:                                          *
 *                                                   *
 * Complete the design of the function called product  *
 * that consumes n double numbers and produces the     *
 * product of those numbers.                         *
 *                                                   *
 * @required the Java "while" loop                   *
 *                                                   *
 * see examples wrapped in check-expect.             *
 *  -with the form: @check_expect (({in}) -> (out))    *
 *                                                   *
 * @author Henrique Rebelo                           *
 *****************************************************/
public class ProductNNumbers2 {

  /**
   * Examples:
   * @check_expect (({new double[] {5}})       -> (5.0))
   * @check_expect (({new double[] {5, 5}})    -> (25.0))
   * @check_expect (({new double[] {-5, 5}})   -> (-25.0))
   * @check_expect (({new double[] {-5, -5}})  -> (25.0))
   * @check_expect (({new double[] {3, 3, 0}}) -> (0.0))
   */
  public static double product(double [] numbers) {
    double product = 1;
    int i = 0; // aux. index for the while loop
         while (i < numbers.length){
            product = numbers[i] * product;
            i++;
         }
    return product;
  }
}
```

Score

5/5

2023-03-12T20:08:25Z

Calling with Arguments

	Name	Arguments	Actual	Expected
pass	sum	0	0	0
pass	sum	1	1	1
pass	sum	10	55	55
pass	sum	-10	0	0
pass	sum	100	5050	5050

Submitted files

SumNatural.java:

```
/*****************************************************************
 *                                                               *
* PROBLEM:                                                       *
*                                                               *
* Complete the design of the function called sum                 *
* that consumes a natural number and produces the                *
* sum of [all] natural numbers starting from 1 to                *
* the consumed x.                                               *
* That is:  1 + 2 + 3 + ... + x                                 *
*                                                               *
* @required the Java "while" loop                               *
*                                                               *
* see examples wrapped in check-expect.                         *
*  -with the form: @check_expect (({in}) -> (out))              *
*                                                               *
* @author Henrique Rebelo                                       *
 *****************************************************************/
public class SumNatural {

  /**
   * Examples:
   * @check_expect (({0})   -> (0))
   * @check_expect (({1})   -> (1))
   * @check_expect (({10})  -> (55))
   * @check_expect (({-10}) -> (0))
   * @check_expect (({100})  -> (5050))
   */
  public static int sum(int x) {
    int sum = 0;
    int count = 1; // first number to start sum
      while (count <= x){
          sum = count + sum;
          count++;
      }
    return sum;
  }
}
```

Score

5/5

2023-03-12T20:06:54Z

Calling with Arguments

	Name	Arguments	Actual	Expected
pass	averageRainfall	new double[] { 1, 2, 3, 4, 5, 9999 }	3.0	3.0
pass	averageRainfall	new double[] { 1, 2, -3, 4, 5, 9999 }	3.0	3.0
pass	averageRainfall	new double[] { 1, 2, 3, 4, 5, 9999, 6, 7 }	3.0	3.0
pass	averageRainfall	new double[] {1, 2, 3, -4, 9999, 5}	2.0	2.0
pass	averageRainfall	new double[] { 10, 9999 }	10.0	10.0
pass	averageRainfall	new double[] { 10, 0, 9999 }	5.0	5.0
pass	averageRainfall	new double[] { -1, -2, -3, 9999 }	0.0	0.0
pass	averageRainfall	new double[] { 9999 }	0.0	0.0

Submitted files

Rainfall.java:

```
/**************************************************
 *                                                *
 * PROBLEM:                                       *
 *                                                *
 * Complete the design of the function called     *
 * averagerainfall that consumes an array of double *
 * and produces the average, but ignore negative values *
 * (which must have been measurement errors), and stop  *
 * when you reach the sentinel 9999.              *
 *                                                *
 * see examples wrapped in check-expect.          *
 *  -with the form: @check_expect (({in}) -> (out))  *
 *                                                *
 * @author Henrique Rebelo                        *
 **************************************************/
public class Rainfall {
  /**
         * Examples:
         * @check_expect (({new double[] {1, 2, 3, 4, 5, 9999}})       -> (3.0))
         * @check_expect (({new double[] {1, 2, -3, 4, 5, 9999}})      -> (3.0))
         * @check_expect (({new double[] {1, 2, 3, 4, 5, 9999, 6, 7}}) -> (3.0))
         * @check_expect (({new double[] {1, 2, 3, -4, 9999, 5}})      -> (2.0))
         * @check_expect (({new double[] {10, 9999}})                  -> (10.0))
         * @check_expect (({new double[] {10, 0, 9999}})               -> (5.0))
         * @check_expect (({new double[] {-1, -2, -3, 9999}})          -> (0.0))
         * @check_expect (({new double[] {9999}})                      -> (0.0))
         */
   public static double averageRainfall(double[] rainfall) {
      double sum = 0;
      int i = 0;
      int count = 0;
      while (rainfall[i] != 9999){
         if (rainfall[i] >= 0){
            sum = rainfall[i] + sum;
            count++;
         }
         i++;

      }
      if (count > 0){
         return sum / count;
      } else {
         return 0;
      }


  }
}
```

Score

8/8

2023-03-12T22:04:44Z

Calling with Arguments

	Name	Arguments	Actual	Expected
pass	getGradesInAscendingOrder	new double [] {10}	[10.0]	[10.0]
pass	getGradesInAscendingOrder	new double [] {10, 0}	[0.0, 10.0]	[0.0, 10.0]
pass	getGradesInAscendingOrder	new double [] {10, 9, 8, 2}	[2.0, 8.0, 9.0, 10.0]	[2.0, 8.0, 9.0, 10.0]
pass	getGradesInAscendingOrder	new double [] {5, 9, 1, 2, 3}	[1.0, 2.0, 3.0, 5.0, 9.0]	[1.0, 2.0, 3.0, 5.0, 9.0]
pass	getGradesInAscendingOrder	new double [] {10, 9, 10, 1, 2, 1}	[1.0, 1.0, 2.0, 9.0, 10.0, 10.0]	[1.0, 1.0, 2.0, 9.0, 10.0, 10.0]

Submitted files

StudentGradesAscendingOrder.java:

```
import java.util.*;

/**************************************************
 *                                                *
 * PROBLEM:                                       *
 *                                                *
 * Complete the design of the function called     *
 * getGradesInAscendingOrder that consumes an array of *
 * grades and produces the grades in ascending order.  *
 *                                                *
 * see examples wrapped in check-expect.          *
 *  -with the form: @check_expect (({in}) -> (out))    *
 *                                                *
 * @author Henrique Rebelo                        *
 **************************************************/
public class StudentGradesAscendingOrder {
        /**
         * Examples:
         * @check_expect (({new double[] {10}})                -> (double[]{10}))
         * @check_expect (({new double[] {10, 0}})             -> (double[]{0, 10}))
         * @check_expect (({new double[] {10, 9, 8, 2}})       -> (double[]{2, 8, 9, 10}))
         * @check_expect (({new double[] {5, 9, 1, 2, 3}})     -> (double[]{1, 2, 3, 5, 9}))
         * @check_expect (({new double[] {10, 9, 10, 1, 2, 1}}) -> (double[]{1, 1, 2, 9, 10, 10}))
         */
        public static double [] getGradesInAscendingOrder(double [] grades) {
                for (int i = 0; i < grades.length; i++){
                   Arrays.sort(grades);
                }
                return grades;
        }
}
```

Score

5/5

2023-03-12T22:08:32Z

Calling with Arguments

	Name	Arguments	Actual	Expected
pass	getGradesInDescendingOrder	new double [] {10}	[10.0]	[10.0]
pass	getGradesInDescendingOrder	new double [] {0, 10}	[10.0, 0.0]	[10.0, 0.0]
pass	getGradesInDescendingOrder	new double [] {2, 8, 9, 10}	[10.0, 9.0, 8.0, 2.0]	[10.0, 9.0, 8.0, 2.0]
pass	getGradesInDescendingOrder	new double [] {5, 9, 1, 2, 3}	[9.0, 5.0, 3.0, 2.0, 1.0]	[9.0, 5.0, 3.0, 2.0, 1.0]
pass	getGradesInDescendingOrder	new double [] {10, 9, 10, 1, 2, 1}	[10.0, 10.0, 9.0, 2.0, 1.0, 1.0]	[10.0, 10.0, 9.0, 2.0, 1.0, 1.0]

Submitted files

StudentGradesDescendingOrder.java:

```
import java.util.*;

/**************************************************
 *                                                *
 * PROBLEM:                                       *
 *                                                *
 * Complete the design of the function called     *
 * getGradesInDescendingOrder that consumes an array of *
 * grades and produces the grades in descending order.  *
 *                                                *
 * see examples wrapped in check-expect.          *
 *  -with the form: @check_expect (({in}) -> (out))     *
 *                                                *
 * @author Henrique Rebelo                        *
 **************************************************/
public class StudentGradesDescendingOrder {

        /**
         * Examples:
         * @check_expect (({new double[] {10}})               -> (double[]{10}))
         * @check_expect (({new double[] {0, 10}})            -> (double[]{10, 0}))
         * @check_expect (({new double[] {2, 8, 9, 10}})      -> (double[]{10, 9, 8, 2}))
         * @check_expect (({new double[] {5, 9, 1, 2, 3}})    -> (double[]{9, 5, 3, 2, 1}))
         * @check_expect (({new double[] {10, 9, 10, 1, 2, 1}}) -> (double[]{10, 10, 9, 2, 1, 1}))
         */
        public static double [] getGradesInDescendingOrder(double [] grades) {

                for (int i = 0; i < grades.length; i++){
                   grades[i] = -grades[i];
                }

                Arrays.sort(grades);
                for (int i = 0; i < grades.length; i++){
                   grades[i] = -grades[i];
                }
                return grades;
        }
}
```

Score

5/5

2023-03-12T22:32:53Z

Calling with Arguments

	Name	Arguments	Actual	Expected
pass	getStringInReverseOrder	new String("Java")	avaJ	avaJ
pass	getStringInReverseOrder	new String("String")	gnirtS	gnirtS
pass	getStringInReverseOrder	new String("12345678910")	01987654321	01987654321

Submitted files

StringReverseOrder.java:

```
/**************************************************
 *                                                *
 * PROBLEM:                                       *
 *                                                *
 * Complete the design of the function called     *
 * getStringInReverseOrder that consumes a String *
 * and produces it in a reverse order.            *
 *                                                *
 * see examples wrapped in check-expect.          *
 *  -with the form: @check_expect (({in}) -> (out))*
 *                                                *
 * @author Henrique Rebelo                        *
 **************************************************/
public class StringReverseOrder {

        /**
         * Examples:
         * @check_expect (({"Java"})        -> ("avaJ"))
         * @check_expect (({"String"})      -> ("gnirtS"))
         * @check_expect (({"12345678910"}) -> ("0198765432"))
         */
        public static String getStringInReverseOrder(String s) {
                String str = "";
                char c;

                for (int i = 0; i < s.length(); i++){
                   c = s.charAt(i);
                   str = c + str;
                }

                return str;
        }
}
```

Score

3/3

2023-03-12T23:13:15Z