Payroll FRQ

SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Assume that the classes listed in the Java Quick Reference have been imported where appropriate. Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

Employees at a store are paid daily wages according to the following rules.

Each employee is paid the same fixed amount per day.
Each employee is paid an additional amount for each item they sold on that day.

Daily Bonus: If the number of items sold that day by an employee is greater than a computed threshold, then the employee also receives a bonus equal to **10** percent of the employee's daily wages.

```
int[] itemsSold; // number of items sold by each employee

double[] wages; // wages to be computed in part (b)

/** Returns the bonus threshold as described in part (a).
 */
/* To be implemented in part (a) */

/** Computes employee wages as described in part (b)
 * and stores them in wages.
 * The parameter fixedWage represents the fixed amount each
 * employee is paid per day.
 * The parameter perItemWage represents the amount each employee
 * is paid per item sold.
 */

/* To be implemented in part (b) */
```

The bonus threshold is calculated based on the number of items sold by all employees on a given day. The employee with the greatest number of sales and the employee with the least number of sales on that day are ignored in the calculation. The average number of items sold by the remaining employees on that day is computed, and this value is used as the bonus threshold.

For a given day, the number of items sold by each employee is stored in the array `itemsSold`. The example below shows the contents of `itemsSold` for a day in which there were ten employees. Each array index represents an individual employee. For example, `itemsSold[3]` represents the number of items sold by employee 3.

itemsSold	0	1	2	3	4	5	6	7	8	9
	48	50	37	62	38	70	55	37	64	60

Based on the information in the table, the bonus threshold is calculated as follows.

((48+50+37+62+38+70+55+37+64+60) - 37 - 70) / 8 = 51.75

(a) Complete a code segment below, which stores the bonus threshold based on the contents of the `itemsSold` array, in a variable and then prints it. Assume that `itemsSold` has been filled appropriately, and that the array contains at least three employees.

```
/** Stores the bonus threshold as described in part (a), in a
 *  variable and then prints it.
 */
```

This code segment is intended to calculate the wages for each employee and to assign them to the appropriate element of the array wages. For example, `wages[3]` should be assigned the wages for employee **3**.

An employee's wages consist of their daily wages plus a possible bonus and are calculated as follows.

Each employee's wages are equal to the fixed wage plus the number of items sold times the amount paid per item sold.
If the employee sold more items than the bonus threshold, the employee also receives a **10** percent bonus added to their wages.

As described in part *(a)*, `computeBonusThreshold()` returns `51.75` for the example array below.

itemsSold	0	1	2	3	4	5	6	7	8	9
	48	50	37	62	38	70	55	37	64	60

Suppose that `fixedWage` is `10.0` and `perItemWage` is `1.5`.

Employee **0** did not sell more items than the bonus threshold, so employee **0**'s wages are equal to `10.0 + 1.5 * 48`, which evaluates to `82.0`. This value will be assigned to `wages[0]`. Employee **9** sold more items than the bonus threshold, so employee **9** receives a **10** percent bonus. Employee **9**'s wages are equal to `(10.0 + 1.5 * 60) * 1.1`, or `110.0`. This value will be assigned to `wages[9]`.

(b) Write a code segment below that computes employee wages as described above and stores them in wages. Assume that `itemsSold` has been filled appropriately, and there are at least three employees in the array. Assume also that the `wages` array and the `itemsSold` array have the same length.

```
/** Computes employee wages as described in part (b)
 *  and stores them in wages.
 *  The parameter fixedWage represents the fixed amount each
 *  employee is paid per day.
 *  The parameter perItemWage represents the amount each employee
 *  is paid per item sold.
 */
double fixedWage, perItemWage;
```