Consider a grade-averaging scheme in which the final average of a student’s scores is computed differently from

the traditional average if the scores have “improved.” Scores have improved if each score is greater than or equal

to the previous score. The final average of the scores is computed as follows.

A student has n scores indexed from 0 to n-1. If the scores have improved, only those scores with indexes

greater than or equal to n/2 are averaged. If the scores have not improved, all the scores are averaged.

The following table shows several lists of scores and how they would be averaged using the scheme described

above.

Student Scores Improved? Final Average

50, 50, 20, 80, 53 No (50 + 50 + 20 + 80 + 53) / 5.0 = 50.6

20, 50, 50, 53, 80 Yes (50 + 53 + 80) / 3.0 = 61.0

20, 50, 50, 80 Yes (50 + 80) / 2.0 = 65.0

Consider the following incomplete StudentRecord class declaration. Each StudentRecord object

stores a list of that student’s scores and contains methods to compute that student’s final average.

public class StudentRecord

{

private int[] scores; // contains scores.length values

// scores.length > 1

// constructors and other data fields not shown

// returns the average (arithmetic mean) of the values in scores

// whose subscripts are between first and last, inclusive

// precondition: 0 <= first <= last < scores.length

private double average(int first, int last)

{ /\* to be implemented in part (a) \*/ }

// returns true if each successive value in scores is greater

// than or equal to the previous value;

// otherwise, returns false

private boolean hasImproved()

{ /\* to be implemented in part (b) \*/ }

// if the values in scores have improved, returns the average

// of the elements in scores with indexes greater than or equal

// to scores.length/2;

// otherwise, returns the average of all of the values in scores

public double finalAverage()

{ /\* to be implemented in part (c) \*/ }

}

(a) Write the StudentRecord method average. This method returns the average of the values in

scores given a starting and an ending index.

Complete method average below.

// returns the average (arithmetic mean) of the values in scores

// whose subscripts are between first and last, inclusive

// precondition: 0 <= first <= last < scores.length

private double average(int first, int last)

(b) Write the StudentRecord method hasImproved.

Complete method hasImproved below.

// returns true if each successive value in scores is greater

// than or equal to the previous value;

// otherwise, returns false

private boolean hasImproved()

(c) Write the StudentRecord method finalAverage.

public double finalAverage()

In writing finalAverage, you must call the methods defined in parts (a) and (b). Assume that these

methods work as specified, regardless of what you wrote in parts (a) and (b).

Complete method finalAverage below.

// of the elements in scores with indexes greater than or equal

// if the values in scores have improved, returns the average

// to scores.length/2;

// otherwise, returns the average of all of the values in scores