

```
private double average(int first, int last)
{
    double num = last - first + 1;
    double sum = 0;
    for (int k = first; k <= last; k++)
        sum += scores[k];
    return sum / num;
}
```

```
private boolean hasImproved()
{
    boolean improved = true;
    for (int k = 0; k < scores.length - 1; k++)
        if (scores[k] > scores[k+1])
            improved = false;
    return improved;
}
```

```
public double finalAverage()
{
    if (hasImproved())
        return average(scores.length/2, scores.length-1)
    else
        return average(0, scores.length-1)
}
```

Part A:	average	3 points
----------------	----------------	-----------------

- +1/2 initialize sum
- +1 loop over scores
 - +1/2 attempt (must reference scores in body)
 - +1/2 correct (from first to last)
- +1/2 add score to sum (in context of loop)
- +1 calculate and return average
 - +1/2 attempt to calculate average
 - +1/2 return correct value
(Check for int division; must be double quotient)

Part B:	hasImproved	3 points
----------------	--------------------	-----------------

- +1 loop over scores
 - +1/2 attempt (must reference scores in body)
 - +1/2 correct (will lose this if index out of bounds)
- +1 compare consecutive scores (in context of loop)
 - +1/2 attempt
 - +1/2 correct
- +1 return correct boolean
 - +1/2 categorize entire array as improved or not improved
(must be in context of comparing consecutive scores)
 - +1/2 correct value returned

Part C:	finalAverage	3 points
----------------	---------------------	-----------------

- +1 call hasImproved()
 - +1/2 attempt
 - +1/2 correct
- +1 return average of last half
 - +1/2 attempt to average half using average
 - +1/2 return correct average (only if improved)
- +1 return average of all
 - +1/2 attempt to average all using average
 - +1/2 return correct average (only if not improved)

Note: Reimplementing code rather than calling available methods results in score of 0 for the portion of part C related to the code reimplementation.