CSC 385 –Assignment 4 –Faster Sorting Algorithms (20 points)

This problem is taken from a hackerrank programming challenge. The problem is tagged as having medium difficulty in hacker rank: (https://www.hackerrank.com/challenges/fraudulent-activity-notifications)

Problem Description—Fraud Detection

HackerLand National Bank has a simple policy for warning clients about possible fraudulent account activity. If the amount spent by a client on a particular day is *greater than or equal to* 2 ×the client's median spending for the last d days, they send the client a notification about potential fraud. The bank doesn't send the client any notifications until they have at least **d** prior days of transaction data.

Given the value of **d** and a client's total daily expenditures for a period of **n** days (where n>d) write a method which returns the number of times the client will receive a notification over all days.

Note: Your algorithm efficiency should not exceed O(nd) where n is the total number of daily expenditures and d is the number of prior day expenditures used for fraud detection.

What you need to do:

To receive full credit, your method must have the following signature:

```
public int getNumberOfFrauds(int[] dailyExpeditures, int d)
```

For example:

`getNumberOfFrauds({4,3,2,2,3,6,8,9,10}, 5)` must return 3

Explanation: We must determine the total number of notifications the client receives over a period of days. For the first five days, the customer receives no notifications because the bank has insufficient transaction data and the number of notifications are zero.
On the sixth day, the bank has 5 days of prior transaction data {4,3,2,2,3} and the median is 3 dollars, the client spent 6 dollars on the sixth day which triggers a notification because 6 >=2* median, so the number of notifications after sixth day is 1.
On the seventh day, the bank has 5 days of prior transaction data {3,2,2,3,6} and the median is 3 dollars, the client spent 8 dollars on the seventh day which triggers a notification because 8 >=2* median, so the number of notifications after seventh day is 2.
On the eights day, the bank has 5 days of prior transaction data {2,2,3,6,8} and the median is 3 dollars, the client spent 9 dollars on the eighth day which triggers a notification because 9>=2* median, so the number of notifications after eighth day is 3.
On the ninth day, the bank has 5 days of prior transaction data {2,3,6,8,9} and the median is 6 dollars, the client spent 10 dollars on the ninth day which does not triggers a notification because 10<2* median, so the number of notifications after ninth day stays at 3.
We then return the final value of notifications which is 3.

Hints: You can store the first d day spending in another array and sort this array using a fast sorting algorithm such as quick sort. For instance, for the above example, the sorted array would be {2,2,3,3,4}. Compare the next day spending (6 dollars) with the median and update the number of frauds. Then replace the first day spending (4 dollars) with d+1 day spending (6 dollars) in the sorted array, that is: {2,2,3,3,6}. If the resulting array is not sorted, shift array elements until it is sorted again. Find the median and compare it with d+2 day spending and update the number of frauds. Next, replace the second day spending (3 dollars) with d+2 day spending (8 dollars) in the sorted array, that is: {2,2,8,3,6}. If the resulting array is not sorted, shift the array elements until it is sorted again {2,2,3,6,8}. Find the median and compare it with d+3 day spending and update the number of frauds. Continue this process until you get to the last day spending and return the number of frauds.

Once you write your program you can test it on hacker Rank against their extensive test cases (https://www.hackerrank.com/challenges/fraudulent-activity-notifications) . If it passes all the test cases on hacker rank then your program is probably correct. It is fun. I had to fix my program about 3-4 times before it could pass very large test cases on hacker rank.

What you need to turn in:

A. Please enclose your **getNumberOfFrauds** method together with all the helper methods and attributes you use in a class called "**FraudDetection.java**". You don't need to include a driver (main method) to test your method. I will test your program against my own test harness or against the hacker rank test cases. **Please note that you cannot use Arrays.sort or any other Java library for sorting. You can; however, use quicksort code from the lectures or textbook.**

B. Please include a document with your submission which briefly describe your algorithm and determine its time efficiency in terms of order of magnitude (Big Oh). Don't just say that my algorithm is O(nd). We already know that. **Explain how you arrived at this time efficiency.**

Grading Rubric

Your program correctly returns the number of frauds and passes all the test cases on hacker	14
Analysis of time efficiency of your algorithm	3
The time efficiency of your algorithm does not exceed O(nd)	3
Total	20

Note: Please do your own work and reach out to me for help if you struggle or have difficulty. If your code found to be copied from the Internet, you will receive no credit for the assignment and the violation will be reported to provost office