Recall the Academic Integrity statement that you signed. Write all answers clearly on these pages, ensuring your final answers are easily recognizable. The number of points for each problem is clearly marked, for a total of 25 points. I will post my solutions on the web on Tuesday, off the Solutions link, after class.

Important: There is a downloadable Eclipse project file that you must do all your work in (see the link on the Weekly Schedule page). Submit all your solutions as Java files online using Checkmate but also turn in this sheet of paper in lab. Recall that this assignment is due on Tuesday morning, May 24 at 11:00am (to give you some more time to write/debug everything).

For 1-5: Each of these methods can be written with one, fairly short loop (and a bit more code). Determine the pattern of scanning/movement in the body of the loop, and then write the for loop bounds to move the appropriate values. Hand simulate a small example before typing your code. Note that none of these methods change the length of the array. Also note that none of these methods will need to check for or throw exceptions. I think they are in increasing order of complexity. Put in stub methods for each so that you can compile/run the driver, testing each new method as you write it.

You may not allocate a second array in these methods: do all the operations in the parameter array.

1. (2 pts) Write a void method named countOccurrences, which takes two parameters: an int[] and an int. It returns the number of times that the int occurs in the int[]; it does not change the array.

2. (2 pts) Write a method named replace, which takes three parameters: an int[] and two int (values). It replaces every occurrence in the array of the first int value with the second int value and returns the number of replacements it made (possibly 0).

3. (2 pts) Write a void method named moveToRear, which takes two parameters: an int[] and an int. It changes the array so that the value at the specified index is moved to the rear; all values following it are moved towards the front by one index. Calling moveToRear with a second argument of 7 means to move the value at index 7 to the rear.

4. (2 pts) Write a void method named moveToFront, which takes two parameters: an int[] and an int. It changes the array so that the value at the specified index is moved to the front; all values preceding it are moved towards the rear by one index. Calling moveToFront with a second argument of 7 means to move the value at index 7 to the front.

5. (2 pts) Write a void method named reverse, which takes one parameter: an int[]. It changes the array, so that its values are reversed: the value at the first index and the value at the last index are exchanged; the values at the second index and the second to last index are exchanged, etc. Hint it is easy to write something that ends up reversing the order twice, leaving the array unchanged: hand simulation will show you your error.
6. (15 pts) You are to complete the unwritten methods in the Portfolio class that manages a portfolio of cash and an arbitrary number of stocks (stored in a Stock[]). Using this program your account starts with a certain amount of cash. You can mutate your portfolio: you can withdraw and deposit cash, buy and sell stock, and update the price of a share of stock. You can also query your portfolio: you can determine how many different stocks you own, how many shares of any particular stock you own, what your portfolio is worth (cash + stocks), whether a certain stock is active (defined by the Stock class), get information about a current stock, and see a String representation of your entire portfolio (all the array values).

First, read and understand the Stock class (an extension of the class used in Quiz #7).

Second, read the comments in the Portfolio class and run my executable (Run Portfolio.bat, which runs PortfolioDriver with my solution to the Portfolio class) in the Quiz8Helper folder, to understand the effect of each of the methods. Enter commands to buy, sell, and update the price of stocks: when prompted, you can make up any small String as a stock’s symbol.

Third, study the methods that I supplied in the Portfolio class to understand how they manipulate its instance variables (cash, stocks, and used). Finally, write the methods required to complete this class.

The methods you need to write are sharesOwned, worth, active, withdraw, sell, and updatePrice. Each of these methods appears as a stub in Portfolio with a specification (as a comment) above it describing how it is supposed to work.

Recall that if we declare Stock[] stocks then stocks[0] (or any index in the stock array) is just like a variable reference to a Stock object (the golden rule of arrays).

sharesOwned

worth

active

withdraw

sell

updatePrice

Important: There is a downloadable Eclipse project file that you must do all your work in (see the link on the Weekly Schedule page). Submit all your solutions as Java files online using Checkmate but also turn in this sheet of paper in lab. Recall that this assignment is due on Tuesday morning, May 24 at 11:00am (to give you some more time to write/debug everything).