1a. (5 points) Add a method to the Array class to return the average of all the numbers in the array. Assume all elements in the array are numbers. So #[1 5.3 4 5 3] would result in the average 3.66. Use either do: or fold:.

1b. (2 points) Write Sunit test(s) for your method in part a.

2a. (5 points) Add a method valuesBetween: a and: b to the Array class. The method returns an Array that contains all the elements of the receiver that are between the values "a" and "b". Use select: in the method.

2b. (2 points) Write Sunit test(s) for your method in part a.

3b. (5 points) Add a method squares to the Collection class. This method returns a collection that contains the squares of the values in the receiver collection. Use collect: to compute the square of each number of a collection. So #(1 2 3) squares returns #(1 4 9).

3b. (2 points) Write Sunit test(s) for your method in part a.

4a. (5 points) Add a method variance to the Collection class. Given the number \(X_1, X_2, \ldots, X_n\) then the variance is \(((X_1 - \text{average}(X))^2 + (X_2 - \text{average}(X))^2 + \ldots + (X_n - \text{average}(X))^2)/(n - 1)\), where \(\text{average}(X)\) is the average of the numbers \(X_1, X_2, \ldots, X_n\). The variance of the numbers 17 15 23 7 9 13 is 33.2.

4b. (2 points) Write Sunit test(s) for your method in part a.

5a. (20 points) Implement a BinarySearchTree class that implements the methods listed below. The search tree should be able to contain any object that implements.

   **add:** - Adds the argument to the树

   **clear:** - Removes all items from the tree

   **size:** - Returns the number of elements in the tree

   **do:** aBlock - Evaluates "aBlock" for each element in the tree

   **withAll:** aCollection - a class method that creates and returns a search tree. The search tree contains the elements in the argument (aCollection).

   **printOn:** aStream - prints a presentation of the search tree. For example:

   ```
   | sample |
   ```
sample := BinarySearchTree new.
sample
    add: 'a';
    add: 'c';
    add: 'b'.
sample printString    returns "BST(a, b, c)"

The elements in the tree are inside parentheses, separated by commas.

5b. (5 points) Write Sunit tests for your BinarySearchTree class.

6. (17 points + 5 points for SUnit tests) Create an Matrix class and SUnit tests for it. An Matrix object holds an N*K matrix of numbers. The class needs to have the following operations:

Matrix class>>rows: numberOfRows columns: numberOfColumns
Returns an Matrix object that has the given number of rows and columns.

Matrix >>row: rowIndex column: columnIndex
Returns the number in the given location

Matrix >>row: rowIndex column: columnIndex put: aNumber
Puts aNumber in the given location.

Matrix >>+ aMatrixObject
Returns a new matrix object that is the sum of the receiver and argument. The receiver and argument need to be of the same size.

Matrix>>* aMatrixObject
Returns a new matrix object that is the product (as matrices) of the receiver and argument.

How to turn in the assignment

In your image create Package called Assignment3. Make sure that all the code for this assignment is in your Assignment3 package. You will be given a store account (source code repository) for the course. You will upload your Assignment3 package to your store account. We will discuss this operation in class.