COP3538
Project 1 – Array Applications

Submission Requirements

- Submit your project folder via the FileUploader tool provided on the website
  - Follow the project submission guidelines for the class

Input File Requirements

- Contest.Input.A.txt (A student data file to load into the program)
- Contest.Input.B.txt (A student data file to load into the program)
- Contest.Input.C.txt (A student data file to load into the program)
- Contest.Description.txt (Describes the layout of the Student.Input.x.txt data file
- Contest.Search.txt (A list of search terms – student names – to load into the program)

Output File Requirements

- Contest.Output.txt (Output file created by the program to store the sorted student data)

Design Specification Requirements

Note: Refer to the sample output in the **Example Output** section below.

1. Create 4 classes (project, Driver, Collection and Student)
   A. The project class (the name will be whatever the project is called)
      1. This is the class that contains the **public static void main(String[] args)** method
      2. This class should only do three things
         a. Display programmer name(s) and project title on separate lines, followed by a blank line
         b. Create an instance of the Driver class
         c. Call the Driver.execute method
   B. The **Driver** class
      1. This class controls the operation of the program starting at the **execute** method
      2. This class must perform the following operations
         a. Compare the 3 sorting algorithms discussed in class
            Note 1: Read in the **Contest.Input.A.txt** input file prior to performing each sorting algorithm.
            Note 2: Use a collection size of 15 to store the contents of the input data file.
            1. Read the data from the input file and store it in a collection object
            2. Sort the collection on the student's full name using the appropriate sorting algorithm
               Note: a student's full name should be like "Smith, John".
               a. Bubble Sort
               b. Selection Sort
               c. Insertion Sort
            3. Capture the number of copies performed by each sorting algorithm
            4. Display a report listing the results of the sorting algorithm (see example output)
         b. Calculate statistics on the data stored in a collection
            Note: Use a collection size of 50 to store the contents of the 3 input data files.
            1. Read the data from the 3 input files and store it in a single collection object
               Note: Create a new collection object.
            2. Sort the collection on the student's total points using the Selection Sort algorithm
            3. Calculate the minimum, maximum, average and median of the total points values
            4. Display a report listing the results of the statistics calculations (see example output)

c. Search the students stored in the collection object
   Note: Use the same collection created for calculating statistics.
   1. Read the **Contest.Search.txt** input file into a 15 element String array
      Note: Use each of the values in this array when searching the collection.
   2. Sort the collection on the student's full name using the Selection Sort algorithm
   3. Search the collection using the Linear Search algorithm
      a. Determine if the search term was found in the collection
      b. If so, determine how many "probes" were required to find the item
   4. Search the collection using the Binary Search algorithm
      a. Determine if the search term was found in the collection
      b. If so, determine how many "probes" were required to find the item
   5. Display a report listing the results of each search algorithms (see example output)
d. Write the collection data to an output file
   Note: Name the output file **Contest.Output.txt**.
   1. Remove each Student object from the collection
   2. Write the Student object data to the output file

C. The **Collection** class
   1. This class contains a n-element array
      Note: The size of the array will be determined when the Collection object is created.
   2. The class provides the following **public** methods
      Note 1: Other methods (public or private) may exist, but the following methods **MUST** exist.
      Note 2: For each method, use the appropriate parameters and return type
      a. add – Adds a Student object to the collection
      b. binarySearch – Searches the collection using the Binary Search algorithm
      c. bubbleSort – Sorts the collection on a **student's full name** using the Bubble Sort algorithm
      d. display – Displays the student object data stored in the collection
      e. getAverage – Calculates the average total points of the students stored in the collection
      f. getMax – Calculates the maximum total points of the students stored in the collection
      g. getMin – Calculates the minimum total points of the students stored in the collection
      h. getMedian – Calculates the median total points of the students stored in the collection
      i. insertionSort – Sorts the collection on a **student's full name** using the Insertion Sort algorithm
      j. isEmpty – Tests whether the collection is empty
      k. linearSearch – Searches the collection using the Linear Search algorithm
      l. remove – Removes a Student object from the collection
      m. selectionSort – Sorts the collection on a **student's full name** using the Selection Sort algorithm
      n. sortArray – Sorts the collection on a **student's total points** using the Selection Sort algorithm

D. The **Student** class
   Note 1: The constructor should **parse** the data into the individual object variables.
   Note 2: Refer to the Contest.Description.txt file for information on parsing the student records.
   1. This class represents a single student
   2. The class provides the following **public** methods
      Note 1: Other methods (public or private) may exist, but the following methods **MUST** exist.
   3. Note 2: For each method, use the appropriate parameters and return type
      a. getFullName – Returns the full name of the student in the format LAST_NAME, FIRST_NAME
      b. getPoints – Returns the total points of the student
      c. toString – Displays the contents of the student object
         Note: The toString method **Overrides** the Object.toString() method.

Additional Notes

- Refer to chapters 2 and 3 of the textbook for assistance with the search and sorting algorithms
- Ensure the source code conforms to the coding standards for the class
- Format the output using the **String.format** method
- Remove as much **redundant (duplicate) code** as possible. If a process must be completed multiple times, try to find a way to write the code only once, but still perform the code multiple times.

Example Output

```
Ima Java Developer
Project 1

Sort Algorithm Report:
The Bubble Sort algorithm required 81 copies
The Selection Sort algorithm required 21 copies
The Insertion Sort algorithm required 27 copies

Student Name         Position  Attempted  Completed  Total Points
------------         --------  ---------  ---------  ------------
Ellis, Linda                6          4          3           500
Fuller, Jonathan            3          9          4           850
// 6 Additional Results
Scott, Emily                9          3          2           250
Warren, Eugene              7          3          3           450

Statistics Report:
The maximum points earned was 1400
The minimum points earned was 150
The average points earned was 746.88
The median points earned was 775.00

Linear Search Results:
Search String        Found  Not Found  # Probes
-------------        -----  ---------  --------
Alvarez, Lawrence        X                   1
Banks, Annie                     X
// 11 Additional Results
Thomas, Wanda                    X
Warren, Eugene           X                  32

Binary Search Results:
Search String        Found  Not Found  # Probes
-------------        -----  ---------  --------
Alvarez, Lawrence        X                   5
Banks, Annie                     X
// 11 Additional Results
Thomas, Wanda                    X
Warren, Eugene           X                   6

Contents of Contest.Output.txt file (do not display on screen):
Alvarez, Lawrence            8          4          4           600
Banks, Christina            13          1          1           150
// 28 Additional Results
Thomas, Daniel              10          3          3           450
Warren, Eugene               7          3          3           450
```