Programming Project 3: Reading and Debugging Code

Due date: Oct. 26, 11:55 PM EST.

You may discuss any of the assignments with your classmates and tutors (or anyone else) but **all work for all assignments must be entirely your own**. Any sharing or copying of assignments will be considered cheating. If you get significant help from anyone, you should acknowledge it in your submission (and your grade will be proportional to the part that you completed on your own).

**You may work with a partner on this assignment.** No groups of more than two students are allowed, though. If you chose to work with a partner, both names should be included in the assignment and both students should submit identical assignments to NYU Classes.

Software developers spend much more time reading and debugging code, than they do writing new code. In this assignment you need to complete several problems that involve reading code from Java API and debugging provided code. You will also be given the results of various sort implementations to figure out which results were generated by which algorithm.

Problem 1: Quicksort partition

The program attached to this assignment QuickSortTester.java contains four different partition methods that can be used with a quicksort implementation. A naive programmer wrote this code to show that all of them are working, even though some of the partition methods result in longer overall running time than others.

Your task is to verify which of the partition methods are correct. For each partition method with a bug, you should explain what the bug is and why it causes a problem. You should also provide a small array (as short as possible, 5-20 elements max) that breaks that particular partition method.

In addition to being buggy, some of the partition methods run for a much longer time than others (well, at least on some arrays they do). Explain that time difference and discuss when it occurs.

Problem 2: ArrayList<E> class

Study ArrayList.java source code file that provides the implementation of the ArrayList class that we have been using. Study both the source code and the documentation and answer the following questions:

1. How many constructors are in the class?

2. The data is stored in an array. What is the name and the type of the array data field? What is the initial capacity of the array? What are the rules for growing the array? Is there a maximum size that this array can grow to?

3. What methods are involved in adding an element to the ArrayList (list all the methods that are called, including the ones from other classes)?

4. Think of how you would remove an element from an array (hint: we discussed it in class). How is your approach different than the one used by the remove method of the ArrayList class?

5. In answering the above questions you encountered the access modifier native. Explain what it means and when it is used.

Please, answer these points in order.
**Problem 3: Arrays.sort() method**

What algorithm is used when you call Arrays.sort(...) passing it an integer array as a parameter? Research the algorithm and provide its description (use the pseudocode in the textbook and the lecture notes as a guide for the amount of details that you need to provide). What checks are made and which special cases are handled before the actual sort happens?

**Problem 4: Which sort is it?**

The graphs in Figures 1, 2 and 3 show the time it took several sorting methods to sort arrays of varying sizes.

Someone replaced the names of algorithms with colors. Your task is to figure out which color correspond to which algorithm. The algorithms used are: insertion sort, Java sort (i.e., the algorithm that you investigated in Problem 3), merge sort, quicksort and selection sort.

The array sizes vary from 10,000 elements to 100,000 in increments of 10,000. All arrays consisted of integers. The graphs were generated using different initial ordering of elements: 1) random ordering, 2) ascending ordering (no duplicates), 3) descending ordering (no duplicates). The result of running each sort is an ascending ordering of elements.

You need to explain the assignment of color to algorithms that you decide on. Your discussion and proper reasoning is worth many more points than the correct assignment.

**Problem 5: What is wrong with the following code?**

**Part A: Binary search**

This is the recursive implementation of a binary search. But it does not always produce correct results. Figure out what is wrong, explain why it is wrong and propose a fix.

```
private static int binarySearch(String A[], String key, int min, int max)
{
    // test if array is empty
    if (max < min)
        // set is empty, so return value showing not found
        return -1;
    else
    {
        // calculate midpoint to cut set in half
        int mid = (min + max) / 2;

        // three-way comparison
        if (A[mid].compareTo(key) == 1)
            // key is in lower half
            return binarySearch(A, key, min, mid - 1);
        else if (A[mid].compareTo(key) == -1)
            // key is in upper half
            return binarySearch(A, key, mid + 1, max);
        else
            // key has been found
            return mid;
    }
}
```
Part B: Insertion sort

The following is a generic implementation of the insertion sort algorithm. Sometimes it runs without any problems. Other times it crashes. Figure out what is wrong, explain why it is wrong and propose a fix.

```java
public void sort(E[] list) {
    for (int i = 1; i < list.length; i++) {
        // insert list[i] into a sorted sublist list[0..i-1] so that
        // list[0..i] is sorted.
        E currentElement = list[i];
        int k;
        for (k = i - 1; list[k].compareTo(currentElement) > 0 && k > 0; k--) {
            list[k + 1] = list[k];
        }
        // Insert the current element into list[k+1]
        list[k + 1] = currentElement;
    }
}
```

Part B: Caesar Cipher

The following program was written by a beginner programmer to provide Caesar Cipher encryption and decryption tool (the actual methods that perform these tasks are not provided). The programmer noticed that the user is never given a chance to enter all the information that they need to enter. Figure out what is wrong, explain why it is wrong and propose a fix.

```java
public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.println("What would you like to do? (E)ncrypt or (D)ecrypt?");
    String tool = input.next();
    tool = tool.toUpperCase();
    if (tool.equals("E") || tool.equals("D")) {
        System.out.println("Enter your encryption key: ");
        int shiftKey = input.nextInt();
        System.out.println("Enter your message: ");
        String userMessage = input.nextLine();
        if (tool.equals("E"))
            System.out.println(encrypt(userMessage,shiftKey));
        if (tool.equals("D"))
            System.out.println(decrypt(userMessage,shiftKey));
    } else {
        System.out.println("ERROR: Invalid choice!");
    }
    input.close();
}
```
Grading

Each problem is worth 20 points.

How and What to Submit

Your solutions should be submitted as a single PDF document (not a text file, not a .doc or .docx document!). Each problem should be answered on a separate page (or pages). Each page should contain your first and last names (or two names if you are working with a partner).

If you wish to use your (one and only) freebie for this project (one week extension, no questions asked), then complete the form at http://goo.gl/forms/YFDVBlscEB before the due date for the assignment. All freebies are due seven days after the original due date and should be submitted to NYU Classes.
Figure 1: Times for arrays in initial random order.
Figure 2: Times for arrays in initial ascending order.
Figure 3: Times for arrays in initial descending order.