CMSC 133 Quiz 3 Worksheet

The next quiz for the course will be on Mon, Apr 6. The following list provides additional information about the quiz.

- The quiz will be posted on Mon, Apr 6, 5 PM (afternoon), and due the same day, Mon, Apr 6, at 6:15 PM (afternoon).
- The quiz will be posted similar to a class project. You will write code in an Eclipse project and submit as usual.
- You can only post clarification questions in Piazza. Debugging questions, why code is not compiling, why is code not passing a test, are invalid questions to post in Piazza.
- **Posting of any kind of code in Piazza, during the quiz period, represents an academic integrity violation and will be reported as such.**
- There is no late submission period, therefore you need to submit often and before Mon, Apr 6 at 6:15 PM (afternoon).
- The quiz will be graded based on submit server tests (release and secret) and code inspection (style).
- You must work by yourself.
- You can use class resources (lecture notes, lecture/lab examples, videos, etc.), but no other resources (e.g., code from the web).
- All submissions must be done via the submit server (no e-mail). The highest scoring submission will be used for grading purposes (you can submit as many times as you want before the deadline).
- Sharing of quiz solutions represents an academic integrity violation and will be reported as such.
- Submissions will be checked with cheating detection software.
- Do not wait until close to 6:15 PM to submit your work. Network problems or submit server overloading are not valid excuses for missing a quiz submission. Plan to submit before 6:15 PM.
- If you are an ADS student: The time frame provided takes into consideration your time allocation. If you need any other assistance, contact your instructor. Ignore this entry if you don't know what an ADS is.
- **Dear Students: it is in your best interest to complete this work by yourself, and following the guidelines provided above. You need to identify which topics you understand and which ones you don't, so you can be successful in CMSC132.**

The following exercises cover the material to be included in this quiz. Solutions to these exercises will not be provided, but you are welcome to discuss your solutions with the TAs or instructor during office hours.

Exercises

1. How many objects are associated with each of the following declarations?

    a. int[] a;
    b. String[] b = new String[3];

2. Can you create an array with 0 elements?

3. Write a static method that returns an array with the elements in the range defined by **startIndex** and **endIndex**. The prototype of the method is:

    public static int[] inRange(int[] a, int startIndex, int endIndex)

4. Write a static method that returns an array with the elements of the array parameter in reverse order. The prototype of the method is **public static int[] reverse(int[] a)**

5. Write a static method that places in the output parameter **answer**, elements in array **a** that have a value less than or equal to **upperLimit**. The prototype of the method is **public static int find(int[] a, int upperLimit, int[] answer)**
    You can assume the **answer** array is large enough to fit the answer. The method will return the number of elements having a value less than or equal to **upperLimit**.

6. Write a Java program that:

    a. Reads integer values and places them into an array named **src.**
    b. Reads integer values and places them into array named **find.**
    c. Prints "Yes" if **find** is a subset of **src,** that is, all elements in **find** appear in **src** (duplicates are fine).

7. Implement a program that reads values from the user, places them into an array, and initializes the second half of the array with values that are double of each of the values provided in the first half. For example, if the user provides the values 10, 40, 25, your program will create an array with the values 10, 40, 25, 20, 80, 50. Use the message "Enter number of elements:" to read the number of values provided by the user (e.g., 3 in the previous example) and "Enter values:" to read each of the values provided by the user (e.g., 10, 40, 25, in the previous example).

8. Implement the method **getInRange** that has the prototype below. The method will initialize the first entry of the output parameter **answer** with the number of array elements in the range specified by **startIndex** and **endIndex** and then it will add the elements in the range after that value. The method will return the average of the elements in the range. For example, given the **data** array {47, 62, 9, 10, 5, 69}, calling **getInRange(data, 2, 4, answer)** will initialize the **answer** array with the values 3, 9, 10, 5 and the method will return 8. The method will not perform any computation and return -1 if **a** or **answer** (or both) are NULL, if **startIndex** is greater than **endIndex,** or the index values are outside of the range defined by 0 and the length of the array – 1.

   public static int getInRange(int[] a, int startIndex, int endIndex, int[] answer)

9. Implement the method **getIndicesNegValues** that has the prototype below. The method will return a **new** array with the **indices** (not the values) of the negative values present (if any) in the **data** array. The new array will have the same length as the **data** array. The method will use the **found** parameter to return the number of entries in the array that are negative (you will use the first entry of the **found** array to store that number). Entries of the new array that are not used will have a value of 0. You can assume both parameters are not null.

   public static int[] getIndicesNegValues(int[] data, int[] found)