CMSC 131 Quiz 4 Worksheet

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

- The quiz will be a written quiz (no computer).
- The quiz will be in lecture.
- Closed book, closed notes quiz.
- Answers must be neat and legible.
- Quiz instructions can be found at http://www.cs.umd.edu/~nelson/classes/utilities/examRules.html

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. **It is recommended that you try these exercises on paper first (without using the computer).**

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). Notice you do not need to print the array; you just need to initialize it with the expected values.

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)