Data Structures

Sheet 1

Deadline : Saturday 9th of March 2019 - 11:59 PM

Provide a Java-code implementation for each of the following requirements.

1. Implement the class **MySpecialArrayUtils** with the following **interfaces**:
   a. `public static void reverse(int[] arr)`
   Reverses the elements of a single dimensional array in place.

   b. `public static int[] sumEvenOdd(int[] arr)`
   Returns sum of the even and the odd elements in a single dimensional array of 2 elements [sumEven, sumOdd] and returns an array of [0, 0] if the input array is empty.

   c. `public static double average(int[] arr)`
   Returns the average of the elements in an array and 0 if the array is empty.
   What would you do to avoid a possible integer overflow?

   d. `public static void moveValue(int[] arr, int val)`
   Move elements that equals to val to the end of the array and the rest of the elements to the start with preserving their relative order.
   Try to solve this problem in place (without using extra memory).

   Example:
   Input: [1,2,3,4,5,6,5, 5, 7, 7], val = 5
   Output: [1, 2, 3, 4, 6, 7, 7, 5, 5, 5]

   e. `public static void transpose(int[][] arr)`
   Transpose a 2d-rectangular array.
   Assume your input is always rectangular, but it might be empty.

2. Write down a java interface to find the nth **Fibonacci number** iteratively.
   Fibonacci Sequence: 0, 1, 1, 2, 3, 5, 8, 13, ...
   What would you do to avoid a possible integer overflow?