Assignment #1
COP-3530, Summer 2019

Release: May 15, 2019 **Due date:** May 24, 2019

Purpose:

This assignment has **4 problems**. The purpose of this assignment is to write efficient algorithms and programs in Java and then calculate the running time complexity of your solutions. In some cases, you will be using the recursive methods to solve in efficient way the problem. This will increase your familiarity with the asymptotic notation and how to apply basic techniques of algorithm analysis. These questions relate to some of the important topics we studied in the Module 1.

Submitting Your Assignment:

- Assignments must be turned in via Canvas.
- Please follow these steps for every assignment:
  1. You are allowed to upload only a single file. Since programs will consist of multiple files, your upload must be a compressed "ZIP" file (with extension **.ZIP**).

     ☞ ***No other kinds of compressed files will be accepted!***
  2. Please name your submission as 1_XXXXXXX.ZIP, where XXXXXXX is your seven digit Panther ID number.
  3. **Before creating the zip file**, make sure your output is correct and your program and output meet all of the published specifications
  4. **Before uploading the zip file**, extract the individual files and examine them to make sure you have included all that are required. Don't forget to include your main/tester method.
  5. Please include the following header for each Java program:

```
/*************************************************************
  Purpose/Description: <a brief description of the program>
  Author's Panther ID: <your Panther ID number>
  Certification:
    I hereby certify that this work is my own and none of it is the work of
    any other person.
*************************************************************/
```

  6. Please make sure that you do not include any other personal information in your submission (besides the **Panther ID** in the name of the **ZIP** file and in the headers of your Java files as explained above). For example, no date of birth or name should be found in the document(s) you submit.
  7. Submissions turned in after the due date and/or which don't meet the established formatting rules will not be accepted.

☞ *Failure to follow these simple directions may result in a loss of credit for the assignment*.

Problem #1: (15 points)

Write a **linear** running time complexity program in **Java** to find all the dominant elements in the given array of integers.

An element is a **dominant element** if is greater than all the elements to its right side. The rightmost element in the array is always a dominant element.

Example:

Input: arr[] = {16, 17, 4, 3, 5, 2}
Output: 17 5 2

Important Notes:

- You must add the main method in your program in order to test your implementation.
- There are no data errors that need to be checked as all the data will be assumed correct.
- You can use the array of the previous example to test your program, however, I suggest that you also use other input arrays to validate the correctness and efficiency of your solution.
- Your program MUST be submitted only in source code form (.java file).
- A program that does not compile or does not run loses all correctness points.

Problem #2: (20 points)

Write a program in **Java** to implement a **recursive boolean** function **verify** that returns **True** if the given array contents remain same when array is reversed, i.e., when last element is equal to the first element, second last element is equal to the second element and so on. Otherwise function **verify** must return **False**.

Examples:

Input: arr[] = {12, 32, 67, 32, 12}
Output: True

Input: arr[] = {1, 2, 3, 4, 5}
Output: False

Important Notes:

- You must add the main method in your program in order to test your implementation.
- There are no data errors that need to be checked as all the data will be assumed correct.
- You can use the array of the previous example to test your program, however, I suggest that you also use other input arrays to validate the correctness and efficiency of your solution.
- Your program MUST be submitted only in source code form (.java file).
- A program that does not compile or does not run loses all correctness points.

(continue on next page)

Problem #3: (40 points)

Given an array A of n distinct integer elements with the following property:

- The first k elements ($0 < k < n-1$) are in strictly increasing sequence followed by the strictly decreasing sequence.

Example:

A = {1, 3, 4, 5, 7, 14, 11, 7, 2, -4, -8}.
It monotonically increases from 1 to 14, then decreases from 14 to -8

Implement a **sub-linear** running time complexity program in **Java** that, given an array with the previous property, determines whether a given integer is in the array.

Important Notes:

- You must add the main method in your program in order to test your implementation.
- There are no data errors that need to be checked as all the data will be assumed correct.
- You can use the array of the previous example to test your program, however, I suggest that you also use other input arrays to validate the correctness and efficiency of your solution.
- Your program MUST be submitted only in source code form (.java file).
- A program that does not compile or does not run loses all correctness points.

Problem #4: (25 points)

Implement a program in **Java** that, given an array of integers, places all positive elements at the end of the array **without changing the relative order of positive and negative elements**. Your program must have **O(n)** running time complexity and **O(n)** auxiliary space complexity.

Example:

Input: arr[] = {1, -1, -3, -2, 7, 5, -11, 6}
Output: -1 -3 -2 -11 1 7 5 6

Important Notes:

- You must add the main method in your program in order to test your implementation.
- There are no data errors that need to be checked as all the data will be assumed correct.
- You can use the array of the previous example to test your program, however, I suggest that you also use other input arrays to validate the correctness and efficiency of your solution.
- Your program MUST be submitted only in source code form (.java file).
- A program that does not compile or does not run loses all correctness points.