COP-3530, Fall 2022

Assignment #1

Purpose:

This assignment has **4 main problems**. These problems relate to some of the important topics covered in **Modules 1 and 2**.

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 **ZIP** file and ***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. Inside **ZIP** folder, there should be a separate folder for each question (i.e. Problem 1, Problem 2, Problem 3, etc.)
  4. **For questions that require Java implementation:**
     - The **.java** file must be inside the corresponding problem folder. DO NOT MIX ANSWERS.
     - ONLY SUBMIT **.JAVA** FILES. DO NOT SUBMIT YOUR WHOLE PROJETC'S FOLDER!!!
     - If is required, each .java file should contain ITS OWN main method at the bottom of the file. Please add only one main method for EACH .java file.
  5. **For written questions:**
     - Submit these files INSIDE the specific problem folder.
     - Each answer MUST be identified. It should be easy to tell which question and subsection you are answering!
     - Written questions must be only in **PDF** format.
  6. 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.
**********************************************************/
```

  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:

You have **two sorted lists of integers**, L_1 and L_2. You know the lengths of each list, L_1 has length N_1 and L_2 has length N_2.

(a) Design an **efficient** algorithm (only pseudocode) to output a sorted list $L_1 \cap L_2$ (the intersection of L_1 and L_2).

(b) If you know that $N_2 > N_1$. What is the running time complexity of your algorithm? Justify.

Important Note:

- For this problem, you don't need to submit any implementation in **Java**. Only the pseudocode of your algorithm is required.
- Pseudocode is a simple way of writing programming code in English. It uses short phrases to write code for programs before you actually create it in a specific language.
- Example of **pseudocode**:

  ```
  Set total to zero
  Set grade counter to one
  While grade counter is less than or equal to ten
        Input the next grade
        Add the grade into the total
  Set the class average to the total divided by ten
  Print the class average.
  ```

- More information about pseudocode in: https://computersciencewiki.org/index.php/Pseudocode

Problem #2:

Given a **sorted array of integer**, A, with possible duplicate elements.

Implement an **efficient** (**sublinear** running time complexity) function in **Java** to find in A, the numbers of occurrences of the input value k.

For example, in an array A = {-1, 2, 3, 5, 6, 6, 6, 9, 10} and k = 6, your program should return 3.

Important Notes:

- For this problem 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.
- 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 #3:

(a) Implement a **sublinear** running time complexity **recursive function** in **Java**

public static long exponentiation(long x, int n)

to calculate x^n.

(b) What is the running time complexity of your function? Justify

(c) Give a number of multiplications used by your function to calculate x^{63}. Justify.

Important Notes:

For item (a):

- 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.
- Your function you can use only the basic arithmetic operators (+, -, *, %, and /).
- 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:

Given an array A of n integers, a leader element of the array A is the element that appears more than half of the time in A.

Using **one stack**, implement in **Java** an **O(n)** running time complexity method

static int leader(int[] A)

to find a leader element and return the index (any) of the leader in A. The program must return -1 if no leader element exists.

Examples:

```
int[] a = {23, 23, 67, 23, 67, 23, 45}; ====> leader(a) = 5
int[] a = {23, 24, 67, 23, 67, 23, 45}; ====> leader(a) = -1
```

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.