Java Programming
CSPC 23
Assignment 3

1. Write a Java program that reads an array of integers and then moves every zero to the right side, i.e., towards the end.

2. Write a Java program to find the k (for given k) largest elements in a given array. Elements in the array can be in any order. Don't forget to check boundary condition for k.

3. Write a Java program to check if a given positive number is a palindrome or not.

4. Write a Java program to check if an array of integers contains three increasing adjacent numbers.

5. Write a Java program to find the number of integers within the range of two specified numbers x & y and that are divisible by another number, p.

6. Write a Java program to accept two strings and test if the second string contains the first one.

7. Write a Java program to read two strings from keyboard and compare them lexicographically.

8. Write a Java program to test if a given string contains the specified sequence of char values.

9. Write a Java program to trim any leading or trailing whitespace from a given string.

10. Write a program that reads an arbitrary number of even integers that are in the range 2 to 100 inclusive and counts how many occurrences of each are entered. Indicate the end of the input by entering -1. After entire input has been processed, print all of the values that were entered by the user along with the number of occurrences.

11. Use a one-dimensional array to solve the following problem:
    Write an application that inputs five numbers, each between 10 and 100, inclusive. As each number is read, display it only if it's not a duplicate of a number already read. Provide for the "worst case" in which all five numbers are different. Use the smallest possible array to solve this problem. Display the complete set of unique values input after the user enters each new value.

12. Write a class called **`NumberOfSixes`** that represents the total number of sixes hit by an IPL team in a given match. The **`NumberOfSixes`** class should contain a single integer as instance data, representing the number of sixes hit. Write a constructor to initialize the number of sixes to zero. Write a method called **`setSix`** that increments the value by one whenever a six is hit, and another method called **`getSix`** that

returns the total number of sixes hit so far. Finally, create a driver class called **`SixTracker`** that creates a few **`NumberOfSixes`** objects and tests their methods.

13. Design and implement a set of classes that define various types of reading material: books, novels, magazines, technical journals, textbooks, and so on. Include data values that describe various attributes of the material, such as the number of pages and the names of the primary characters. Include methods that are named appropriately for each class and that print an appropriate message. Create a main driver class to instantiate and exercise several of the classes.

14. Create class **`SavingsAccount`**. Use a static variable **`annualInterestRate`** to store the annual interest rate for all account holders. Each object of the class contains a private instance variable **`savingsBalance`** indicating the amount the saver currently has on deposit. Provide method **`calculateMonthlyInterest`** to calculate the monthly interest by multiplying the **`savingsBalance`** by **`annualInterestRate`** divided by 12—this interest should be added to **`savingsBalance`**. Provide a static method **`modifyInterestRate`** that sets the **`annualInterestRate`** to a new value. Write a program to test class **`SavingsAccount`**. Instantiate two **`savingsAccount`** objects, **`saver1`** and **`saver2`**, with balances of `2000.00 and `3000.00, respectively. Set **`annualInterestRate`** to 4%, then calculate the monthly interest for each of 12 months and print the new balances for both savers. Next, set the **`annualInterestRate`** to 5%, calculate the next month's interest and print the new balances for both savers.

15. Design a class named **`StopWatch`**. The class contains:
    - Private data fields **`startTime`** and **`endTime`** with getter methods.
    - A no-arg constructor that initializes **`startTime`** with the current time.
    - A method named **`start()`** that resets the startTime to the current time.
    - A method named **`stop()`** that sets the endTime to the current time.
    - A method named **`getElapsedTime()`** that returns the elapsed time for the stopwatch in milliseconds.

    Draw the UML diagram for the class and then implement the class in Java.