Class 10 project
Second semester 2025-26
Questions on Arrays

1. Define a single dimensional array of integers in Java to hold 20 elements. Show how you would initialize it with the values 1 to 20 and then display all the values.
2. Write a Java program to accept **n** integer values from the user (via scanner) into an array and then find and print the **maximum** and **minimum** elements in that array.
3. Given an integer array `arr[]` of size `n`, write a Java method `public static int sumEven(int arr[])` that returns the sum of all even numbers in that array. Also show how you would call this method in `main`.
4. Explain how a **two dimensional** array differs from a single dimensional array. Then write a Java program to read a 3×3 `int` matrix from user input and display the transpose of that matrix.
5. Given a 2D array `int a[ ][ ]` of size `m × n`, write code to compute the sum of all elements in it, and also to find the sum of each row and each column.
    - **Sum of Principal (Left) Diagonal**
      Given a 2D array `int a[ ][ ]` of size `m × m` (square matrix), write a program to compute the sum of the **principal diagonal** elements (from `a[0][0]` to `a[m-1][m-1]`).
    - **Sum of Secondary (Right) Diagonal**
      Accept a square matrix `a[ ][ ]` of order `m`. Find and print the sum of the **secondary diagonal** elements (from `a[0][m-1]` to `a[m-1][0]`).
6. **Sum of Both Diagonals**
   Write a program to calculate and print separately:
7. the sum of the principal diagonal,
8. the sum of the secondary diagonal,
9. and the total sum of both (avoiding double counting if `m` is odd and the center element lies on both diagonals).
    - **Sum of Elements Above Principal Diagonal**
      Accept a square matrix `a[ ][ ]`. Find and display the sum of all elements **above** the principal diagonal (`i < j`).
    - **Sum of Elements Below Principal Diagonal**
      Accept a square matrix `a[ ][ ]`. Find and display the sum of all elements **below** the principal diagonal (`i > j`).
    - **Sum of Elements Above Secondary Diagonal**
      For a square matrix `a[ ][ ]`, compute the sum of all elements that lie **above** the secondary diagonal (`i + j < m-1`).
    - **Sum of Elements Below Secondary Diagonal**
      For a square matrix `a[ ][ ]`, compute the sum of all elements that lie **below** the secondary diagonal (`i + j > m-1`).
    - **Sum of Boundary Elements**
      Given a square matrix `a[ ][ ]`, calculate the sum of all **boundary elements** (elements lying in the first row, last row, first column, or last column).
    - **Sum of Non-boundary Elements**
      Accept a square matrix and compute the sum of all elements that are **not boundary elements** (i.e., the inside matrix).
    - **Sum of Corner Elements**
      Write a program to find the sum of the **four corner elements** of a square matrix.

10. Suppose you have `int[] arr = {5, 3, 8, 1, 9}`. Write code to sort this array in ascending order (you may use simple sorting like bubble sort). Print the sorted array.
11. Write a Java program that searches for a given integer `key` in an integer array using **binary search** (assuming the array is already sorted). If found, output the index; otherwise output "Not found".
12. Write a method `public static int countOccurrence(int arr[], int key)` which returns how many times `key` appears in the array `arr`. Test it with an array and key of your choice.
13. A school tracks marks of 5 students in 4 subjects in a 2D array `marks[5][4]`. Write a program to compute and display (a) total marks per student, (b) average marks per student (rounded), and (c) subject-wise totals (i.e. sum in each column).
14. Write a Java program that merges two sorted integer arrays into a **third** sorted array. For example, given `{1, 4, 7}` and `{2, 3, 5, 8}`, the merged sorted array should be `{1, 2, 3, 4, 5, 7, 8}`.
15. **Merging two sorted arrays**
    You are given two sorted integer arrays:
    `a[] = {2, 5, 9, 15}` and `b[] = {1, 3, 12, 14, 20}`.
    Write a Java program (or method) to merge these two arrays into a new sorted array `c[]`, and then display the contents of `c[]`.
16. **Splitting / Partitioning an array into two arrays**
    A teacher has marks of 20 students in one array `marks[20]`. Write a program to split this into two arrays:
    - `high[]` containing marks ≥ 50
    - `low[]` containing marks < 50
      Then display the `high[]` and `low[]` arrays' contents and counts.
17. **Selection Sort**
    Given an unsorted integer array `arr[] = {45, 12, 78, 3, 56}`.
    (a) Use **selection sort** (in ascending order) to sort it.
    (b) Show the array after each pass (i.e. after placing each smallest element).
    (c) Write a method `selectionSort(int arr[])` and call it from `main`.
18. **Finding hottest and coldest city**
    You have two parallel arrays:
    `city[] = {"Delhi", "Mumbai", "Kolkata", "Chennai", "Lucknow"}`
    `temp[] = {40, 35, 42, 38, 36}` (temperature in °C)
    Write a Java program to determine and print:
    - the city with the highest temperature and its temperature
    - the city with the lowest temperature and its temperature
19. **Merging / concatenation of two arrays with duplicates removal**
    Two integer arrays: `a[] = {1, 3, 5, 7}` and `b[] = {3, 4, 5, 8, 9}`.
    Write a program to merge them into one array `c[]` such that each element appears only once.
    Then display `c[]` sorted.
20. **Split into even/odd and sort each**
    Given `arr[] = {12, 5, 18, 7, 10, 3}`.
    (a) Separate even numbers into `even[]` and odd numbers into `odd[]`.
    (b) Sort both arrays in ascending order.
    (c) Print `even[]` and `odd[]`.
21. **Merging descending sorted arrays**
    You have two arrays sorted in **descending** order:
    `a[] = {100, 90, 75, 60}` and `b[] = {95, 85, 50, 40}`
    Write a Java program to merge them into `c[]` so that `c[]` is also in descending order.

22. **Selection sort in descending order & count swaps**
    Given `arr[] = {25, 10, 30, 5, 20}`
    (a) Sort it in **descending order** using selection sort.
    (b) Also count how many swaps are performed.
    (c) Print the sorted array and the swap count.
23. **Finding second highest & second lowest**
    Given an integer array `arr[]` of size `n` (with all distinct elements).
    Write a program to find and print the **second highest** and **second lowest** elements (without sorting the whole array).
24. **Merge and then split based on condition**
    Suppose you have two arrays `a[]` and `b[]` of marks (0–100) from two classes.
    (a) Merge them into `all[]`
    (b) From `all[]`, split into `passed[]` (marks ≥ 35) and `failed[]` (marks < 35)
    (c) Print count and elements of `passed[]` and `failed[]`.

Questions on String Handling

1. Explain the difference between `String` and `StringBuffer`. Give at least two methods used by each.
2. Write a Java program to accept a `String s` from user and print:
    a. length of the string
    b. the first character
    c. the last character
    d. the string in uppercase
    e. the string after trimming spaces at both ends
3. Write a Java method `public static boolean isPalindrome(String s)` that returns `true` if the given string is a palindrome (ignoring case), else `false`. For example, "Level" → true.
4. Given a sentence (String) containing several words separated by spaces, write a Java program to find and print the **longest word** and its length.
5. Write a Java program to accept two strings `s1` and `s2` and check whether they are **anagrams** of each other (i.e. same letters in different order). E.g. "LISTEN" and "SILENT".
6. Write code to replace all occurrences of character `'a'` with `'*'` in a given string `s`, and display the new string.
7. Given a string `s`, write a method `public static String removeVowels(String s)` that returns a new string formed by removing all vowels (AEIOU / aeiou) from `s`.
8. Write a Java program that accepts a full name (first, middle, last) in one line (e.g. "Raj Kumar Sharma") and prints the initials (e.g. "R K S").
9. Write a Java program to accept a sentence and display the words in **reverse order**.
    Example: Input: "COMPUTER IS FUN" → Output: "FUN IS COMPUTER"
10. Write a Java program to count the frequency of each character in a given string (ignoring case). E.g. input "ABBAA" → A:3, B:2.
    i. **Word Splitting**
    Write a program to input a sentence and split it into words. Store the words in a string array and display each word on a new line.
    ii. **Sorting Words Alphabetically**
    Accept a sentence from the user and display the words in **alphabetical order**.
    Example:
    Input: `"MANGO IS A FRUIT"`
    Output: `A FRUIT IS MANGO`.

11. **Longest and Shortest Word**
    Input a sentence and find:
12. the longest word and its length
13. the shortest word and its length.
14. **Frequency of a Word**
    Accept a sentence and a word from the user. Count and print how many times that word appears in the sentence (case insensitive).
15. **Count of Vowels, Consonants, Digits, Special Characters**
    Write a program that accepts a string and prints the number of vowels, consonants, digits, and special characters in it.
16. **Toggle Case of Each Character**
    Accept a string and convert each lowercase letter to uppercase and each uppercase letter to lowercase. Example: `"JaVa"` → `"jAvA"`.
17. **Check Pangram**
    A pangram is a sentence containing every letter of the alphabet at least once. Example: `"The quick brown fox jumps over a lazy dog"`.
    Write a program to accept a string and check whether it is a pangram.
18. **Remove Extra Spaces**
    Input a sentence that may have multiple spaces between words. Output the same sentence with **only single spaces** between words. Example:
    Input: `"JAVA IS FUN"`
    Output: `"JAVA IS FUN"`.
19. **Word with Maximum Frequency**
    Accept a sentence and find the word that occurs most frequently. Display that word and its count.
20. **Palindrome Words in a Sentence**
    Accept a sentence and display all the words which are palindromes (case insensitive). Example: `"MOM AND DAD WENT OUT"` → Palindromes: `MOM, DAD`.
21. **Extract Initials**
    Input a full name (e.g. `"Ravi Shankar Prasad"`) and display initials with surname (e.g. `"R.S. Prasad"`).
22. **Check Substring**
    Accept two strings from the user and check whether the second string occurs as a substring of the first. Display the result.
23. **Reverse Words**
    Accept a sentence and reverse each word **individually** without changing their order.
    Example: `"JAVA IS FUN"` → `"AVAJ SI NUF"`.
24. **Count Words Starting with a Vowel**
    Accept a sentence and count how many words start with a vowel (a, e, i, o, u).
25. **Delete a Word**
    Accept a sentence and a word from the user. Delete all occurrences of that word from the sentence and print the new sentence.