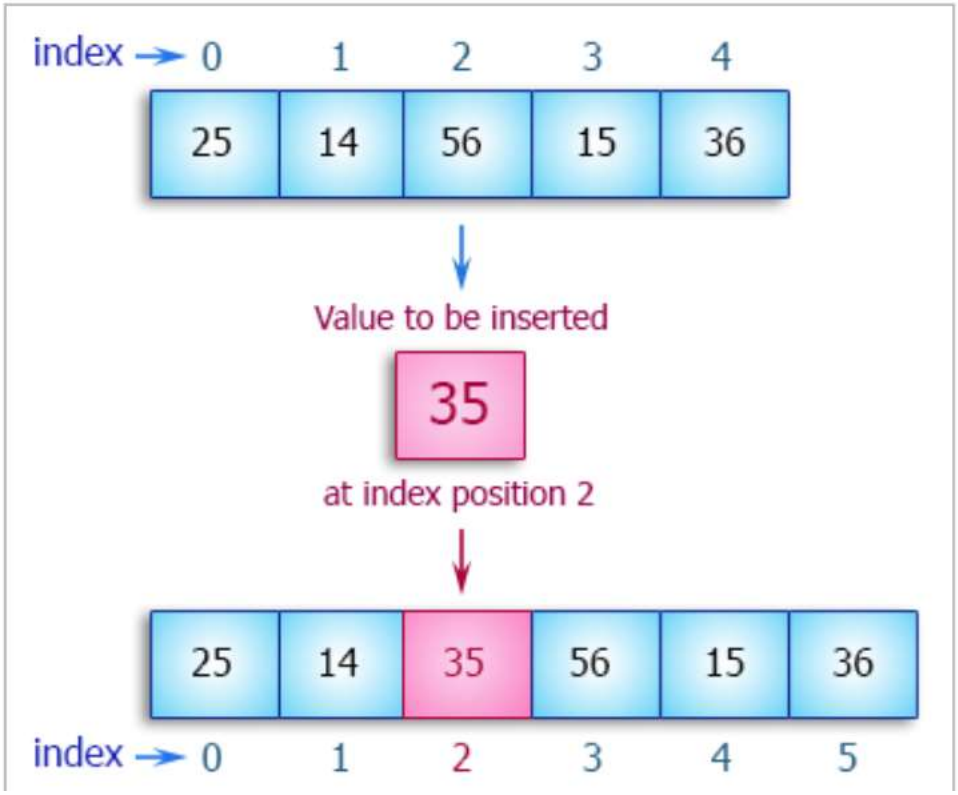
Tutorial: Arrays

Q1. Write a Java program to insert an element (specific position) into an array.

Pictorial Presentation:

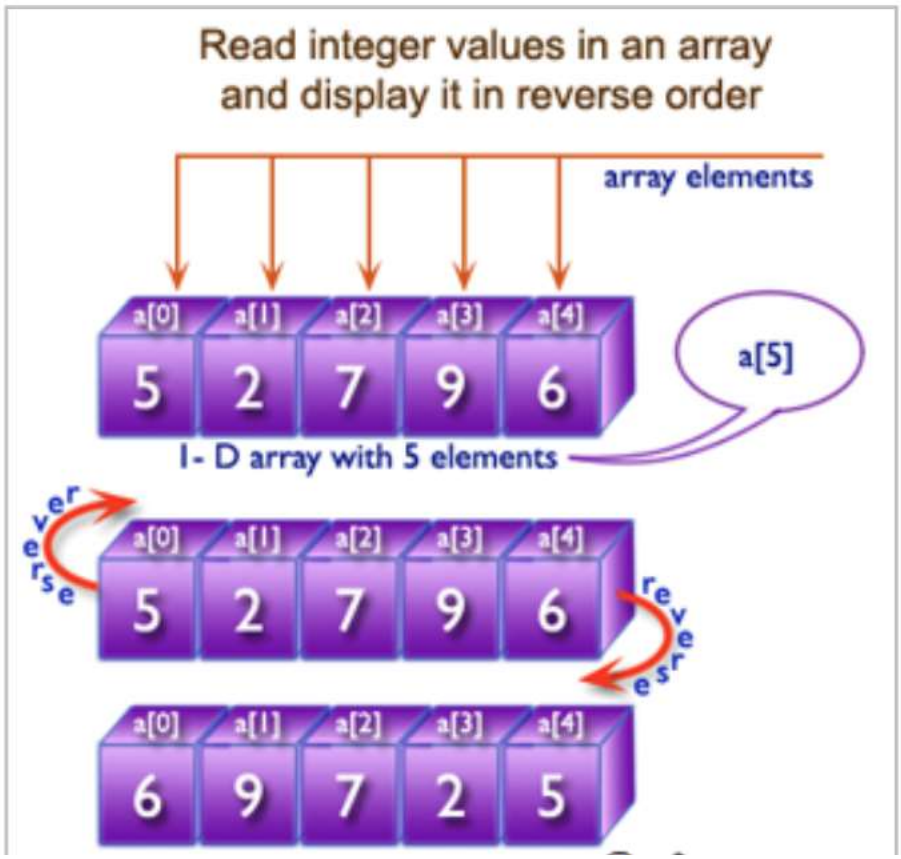


Solution Q1

```java
// Import the Arrays class from the java.util package.
import java.util.Arrays;

// Define a class named Q1.
public class Exercise9 {

    // The main method where the program execution starts.
    public static void main(String[] args) {
        // Declare and initialize an integer array 'my_array'.
        int[] my_array = {25, 14, 56, 15, 36, 56, 77, 18, 29, 49};

        // Define the position where the new element will be inserted.
        int Index_position = 2;

        // Define the value of the new element to be inserted.
        int newValue = 5;

        // Print the original array using Arrays.toString() method.
        System.out.println("Original Array : " +
Arrays.toString(my_array));

        // Loop to shift elements to make space for the new element.
        for (int i = my_array.length - 1; i > Index_position; i--) {
            my_array[i] = my_array[i - 1];
        }

        // Insert the new element at the specified position.
        my_array[Index_position] = newValue;

        // Print the modified array with the new element.
        System.out.println("New Array: " + Arrays.toString(my_array));
    }
}
```

Q2. Write a Java program to reverse an array of integer values.

Pictorial Presentation:

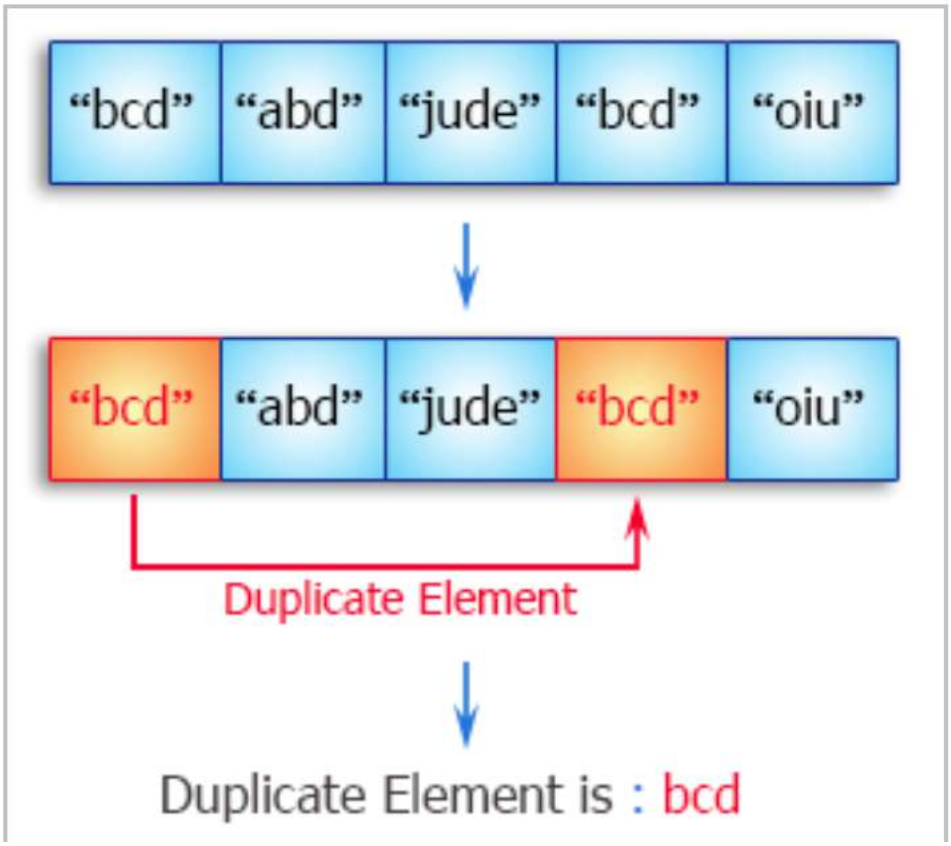


Solution Q2

```java
// Import the Arrays class from the java.util package.
import java.util.Arrays;

// Define a class named Q2.
public class Q2 {

    // The main method where the program execution starts.
    public static void main(String[] args) {
        // Declare and initialize an integer array 'my_array1'.
        int[] my_array1 = {
            1789, 2035, 1899, 1456, 2013,
            1458, 2458, 1254, 1472, 2365,
            1456, 2165, 1457, 2456
        };

        // Print the original array using Arrays.toString() method.
        System.out.println("Original array : " +
Arrays.toString(my_array1));

        // Iterate through the first half of the array and reverse its
elements.
        for (int i = 0; i < my_array1.length / 2; i++) {
            // Swap the elements at positions 'i' and 'length - i -
1'.
            int temp = my_array1[i];
            my_array1[i] = my_array1[my_array1.length - i - 1];
            my_array1[my_array1.length - i - 1] = temp;
        }

        // Print the reversed array using Arrays.toString() method.
        System.out.println("Reverse array : " +
Arrays.toString(my_array1));
    }
}
```

Q3. Write a Java program to find duplicate values in an array of string values.

Pictorial Presentation:

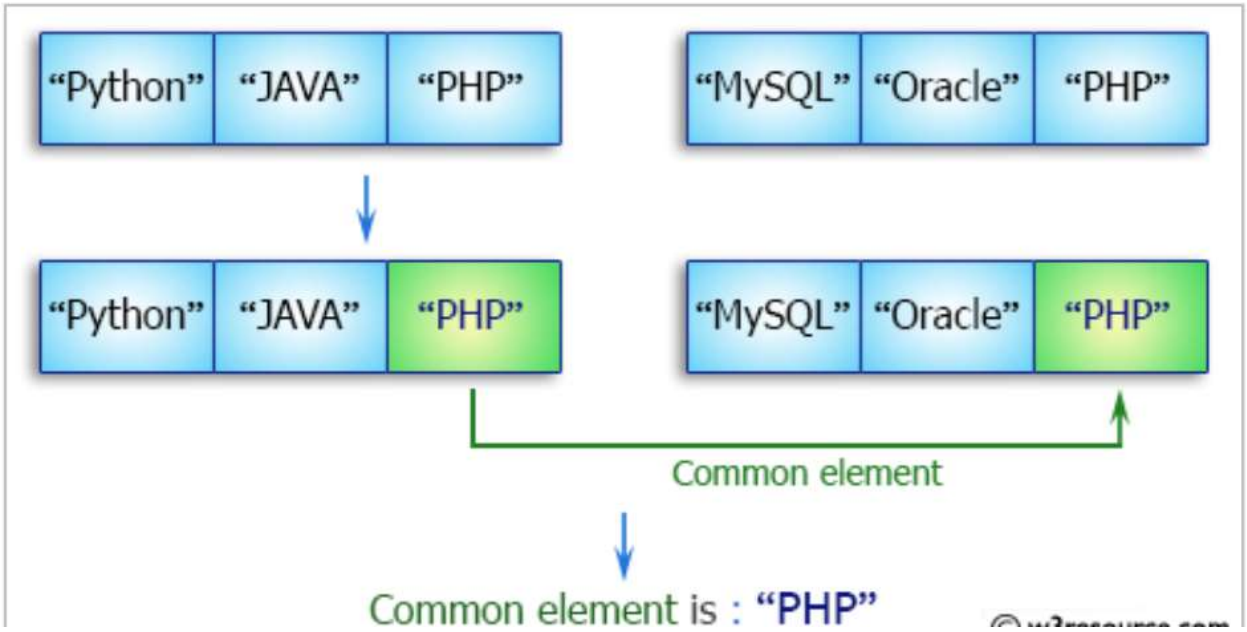


Solution Q3

```java
// Define a class named Q3.
public class Q3 {
    // The main method where the program execution starts.
    public static void main(String[] args) {
        // Declare and initialize a string array 'my_array'.
        String[] my_array = {"bcd", "abd", "jude", "bcd", "oiu",
"gzw", "oiu"};

        // Iterate through the elements of the string array.
        for (int i = 0; i < my_array.length-1; i++) {
            for (int j = i+1; j < my_array.length; j++) {
                // Check if two string elements are equal and not the
same element.
                if ((my_array[i].equals(my_array[j])) && (i != j)) {
                    // If a duplicate is found, print the duplicate
element.
                    System.out.println("Duplicate Element is : " +
my_array[j]);
                }
            }
        }
    }
}
```

Q4. Write a Java program to find common elements between two arrays (string values).

Pictorial Presentation:

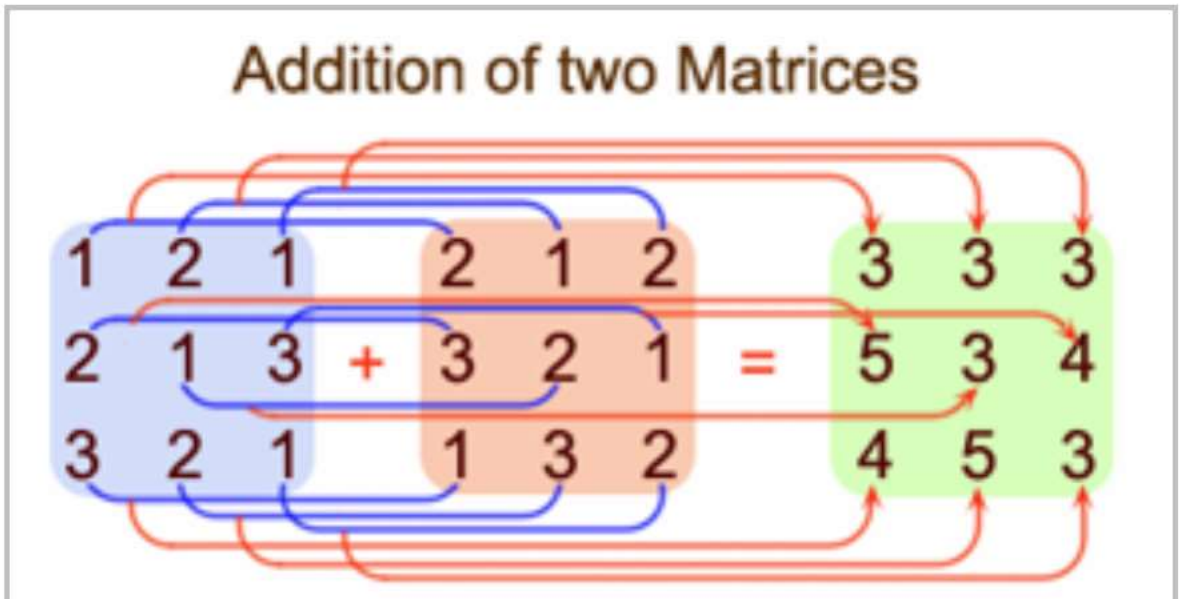


Solution Q4

```java
// Import the necessary Java utilities package.
import java.util.*;

// Define a class named Q4.
public class Q4 {
    // The main method where the program execution starts.
    public static void main(String[] args) {
        // Declare and initialize two string arrays, array1 and
array2.
        String[] array1 = {"Python", "JAVA", "PHP", "C#", "C++",
"SQL"};
        String[] array2 = {"MySQL", "SQL", "SQLite", "Oracle",
"PostgreSQL", "DB2", "JAVA"};

        // Print the original contents of array1 and array2.
        System.out.println("Array1 : " + Arrays.toString(array1));
        System.out.println("Array2 : " + Arrays.toString(array2));

        // Create a HashSet to store common elements.
        HashSet set = new HashSet();

        // Iterate through both arrays to find and store common
elements.
        for (int i = 0; i < array1.length; i++) {
            for (int j = 0; j < array2.length; j++) {
                // Check if elements in array1 and array2 are equal.
                if (array1[i].equals(array2[j])) {
                    // Add the common element to the HashSet.
                    set.add(array1[i]);
                }
            }
        }

        // Print the common elements.
        System.out.println("Common element : " + (set)); // OUTPUT:
[SQL, JAVA]
    }
}
```

Q5. Write a Java program to add two matrices of the same size.

Pictorial Presentation:



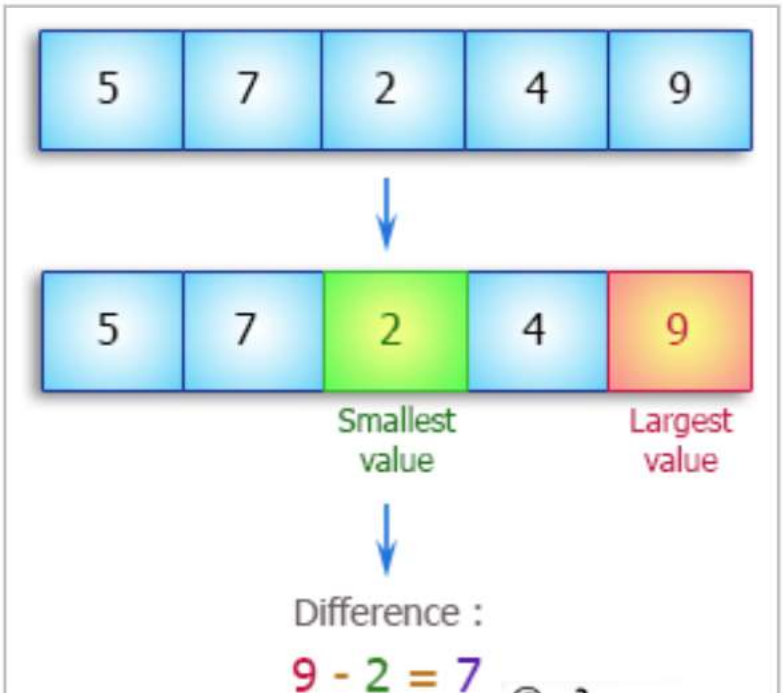
Solution Q5

```java
// Import the Java utility for reading input.
import java.util.Scanner;

// Define a class named Q5.
public class Q5 {
    public static void main(String args[]) {
        int m, n, c, d;

        // Create a new Scanner object to read user input.
        Scanner in = new Scanner(System.in);

        // Prompt the user to input the number of rows for the matrix.
        System.out.println("Input number of rows of the matrix");
        m = in.nextInt();

        // Prompt the user to input the number of columns for the
matrix.
        System.out.println("Input number of columns of the matrix");
        n = in.nextInt();

        // Create two-dimensional arrays to store matrix data.
        int array1[][] = new int[m][n];
        int array2[][] = new int[m][n];
        int sum[][] = new int[m][n];

        // Prompt the user to input elements of the first matrix.
        System.out.println("Input elements of the first matrix");
        for (c = 0; c < m; c++) {
            for (d = 0; d < n; d++) {
                array1[c][d] = in.nextInt();
            }
        }

        // Prompt the user to input elements of the second matrix.
        System.out.println("Input elements of the second matrix");
        for (c = 0; c < m; c++) {
            for (d = 0; d < n; d++) {
                array2[c][d] = in.nextInt();
            }
        }
```

```
        // Calculate the sum of the matrices.
        for (c = 0; c < m; c++) {
            for (d = 0; d < n; d++) {
                sum[c][d] = array1[c][d] + array2[c][d];
            }
        }

        // Display the result, which is the sum of the matrices.
        System.out.println("Sum of the matrices:");
        for (c = 0; c < m; c++) {
            for (d = 0; d < n; d++) {
                System.out.print(sum[c][d] + "\t");
            }
            System.out.println();
        }
    }
}
```

Q6. Write a Java program to get the difference between the largest and smallest values in an array of integers. The array must have a length of at least 1.

Pictorial Presentation:

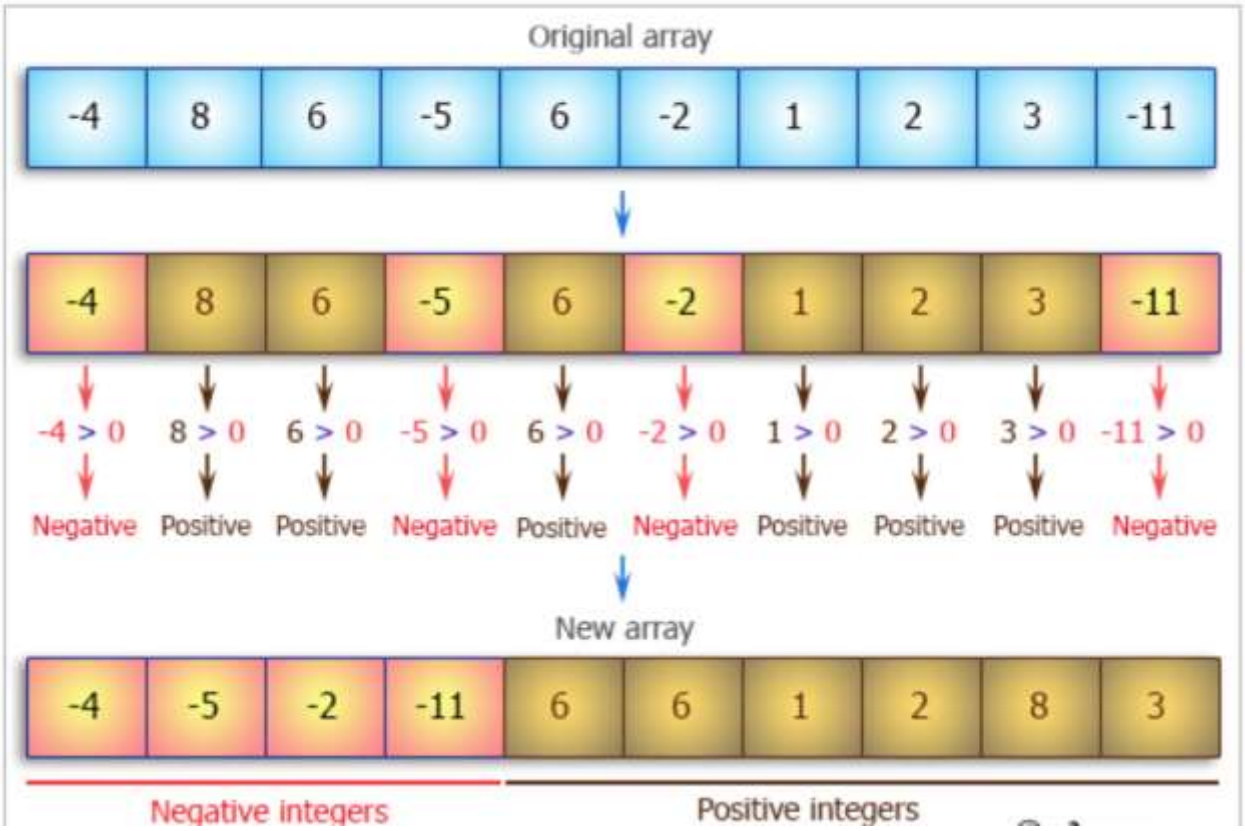


Solution Q6

```java
// Import the java.util package to use utility classes, including
Arrays.
import java.util.Arrays;

// Define a class named Q6.
public class Q6 {
    // The main method for executing the program.
    public static void main(String[] args) {
        // Declare and initialize an array of integers.
        int[] array_nums = {5, 7, 2, 4, 9};

        // Print the original array.
        System.out.println("Original Array: " +
Arrays.toString(array_nums));

        // Initialize variables to store the maximum and minimum
values.
        int max_val = array_nums[0];
        int min = array_nums[0];

        // Use a loop to find the maximum and minimum values in the
array.
        for (int i = 1; i < array_nums.length; i++) {
            if (array_nums[i] > max_val)
                max_val = array_nums[i];
            else if (array_nums[i] < min)
                min = array_nums[i];
        }

        // Calculate and print the difference between the largest and
smallest values.
        System.out.println("Difference between the largest and
smallest values of the said array: " + (max_val - min));
    }
}
```

Q7. Write a Java program to arrange the elements of an array of integers so that all negative integers appear before all positive integers.

Pictorial Presentation:



Solution Q7

```java
// Import the necessary Java utility class for working with arrays.
import java.util.Arrays;

// Define the Main class.
public class Main {

    // The main method for executing the program.
    public static void main(String[] args) {
        // Define an array of integers.
        int[] nums = {-4, 8, 6, -5, 6, -2, 1, 2, 3, -11};

        // Print the original array.
        System.out.println("Original Array: " +
Arrays.toString(nums));

        // Call the sort_nums method to modify the array.
        sort_nums(nums);

        // Print the modified array.
        System.out.println("New Array: " + Arrays.toString(nums));
    }

    // Method to sort the numbers in the array based on their signs.
    public static void sort_nums(int[] nums){
        int pos_num = 0;
        int neg_num = 0;
        int i, j;
        int max = Integer.MIN_VALUE;

        // Count the positive and negative numbers and find the
maximum value.
        for(i = 0; i < nums.length; i++){
            if(nums[i] < 0) neg_num++;
            else pos_num++;
            if(nums[i] > max) max = nums[i];
        }
        max++;

        // If there are no negative or positive numbers, return.
        if(neg_num == 0 || pos_num == 0) return;

        i = 0;
        j = 1;
```

```java
        // Reorder the array.
        while(true){
            while(i <= neg_num && nums[i] < 0) i++;
            while(j < nums.length && nums[j] >= 0) j++;

            if(i > neg_num || j >= nums.length) break;

            nums[i] += max * (i + 1);
            swap_nums(nums, i, j);
        }

        i = nums.length - 1;

        // Adjust the values to their original range.
        while(i >= neg_num){
            int div = nums[i] / max;

            if(div == 0) i--;
            else{
                nums[i] %= max;
                swap_nums(nums, i, neg_num + div - 2);
            }
        }
    }

    // Method to swap elements in the array.
    private static void swap_nums(int[] nums, int i, int j){
        int t = nums[i];
        nums[i] = nums[j];
        nums[j] = t;
    }
}
```

Q8. Write a Java program to find the maximum product of two integers in a given array of integers.

nums = { 2, 3, 5, 7, -7, 5, 8, -5 }

Solution Q8

```
// Import necessary Java classes.
import java.util.*;

// Define a class named 'solution'.
class solution
{
    // A method to find the pair of elements with the maximum product.
    public static void find_max_product(int[] nums)
    {
        int max_pair_product = Integer.MIN_VALUE;
        int max_i = -1, max_j = -1;

        // Loop through the array elements.
        for (int i = 0; i < nums.length - 1; i++)
        {
            for (int j = i + 1; j < nums.length; j++)
            {
                // Check if the product of elements at indices i and j
is greater than the current maximum.
                if (max_pair_product < nums[i] * nums[j])
                {
                    max_pair_product = nums[i] * nums[j];
                    max_i = i;
                    max_j = j;
                }
            }
        }

        // Print the pair and maximum product.
        System.out.print("Pair is (" + nums[max_i] + ", " +
nums[max_j] + "), Maximum Product: " + (nums[max_i] * nums[max_j]));
    }

    public static void main (String[] args)
    {
        int[] nums = { 2, 3, 5, 7, -7, 5, 8, -5 };
        System.out.println("\nOriginal array:
"+Arrays.toString(nums));

        // Find and print the pair of elements with the maximum
product.
        find_max_product(nums);
    }
```

```
}
```