11TH SEPTEMBER ASSIGNMENT

Q1. Question: Given an array where all its elements are sorted in increasing order except two swapped elements, sort it in linear time.

Answer in Java:

java

Copy code

```
public static void sortWithTwoSwaps(int[] arr) {
    int firstSwapIndex = -1;
    int secondSwapIndex = -1;

    // Find the two elements that are swapped
    for (int i = 0; i < arr.length - 1; i++) {
        if (arr[i] > arr[i + 1]) {
            if (firstSwapIndex == -1) {
                firstSwapIndex = i;
            } else {
                secondSwapIndex = i + 1;
                break;
            }
        }
    }

    // Swap the elements back to their correct positions
    if (firstSwapIndex != -1 && secondSwapIndex != -1) {
        int temp = arr[firstSwapIndex];
        arr[firstSwapIndex] = arr[secondSwapIndex];
        arr[secondSwapIndex] = temp;
    }
}
```

Q2. Question: Given an array of positive and negative integers, segregate them in linear time and constant space. The output should print all negative numbers, followed by all positive numbers.

Answer in Java:

java

Copy code

```
public static void segregatePositiveNegative(int[] arr) {
    int n = arr.length;
    int left = 0;
    int right = n - 1;

    while (left <= right) {
        if (arr[left] < 0 && arr[right] < 0) {
            left++;
        } else if (arr[left] >= 0 && arr[right] < 0) {
            // Swap arr[left] and arr[right]
            int temp = arr[left];
            arr[left] = arr[right];
            arr[right] = temp;
            left++;
            right--;
        } else {
            right--;
        }
    }
}
```

Q3. Question: Given an array of positive and negative integers, segregate them in linear time and constant space. The output should print all negative numbers, followed by all positive numbers. The relative order of elements must remain the same.

Answer in Java:

java

Copy code

```
public static void segregatePositiveNegativeSameOrder(int[] arr) {
    int n = arr.length;
    int negIndex = 0;
```

```
    for (int i = 0; i < n; i++) {
        if (arr[i] < 0) {
            // Swap arr[i] with arr[negIndex]
            int temp = arr[i];
            arr[i] = arr[negIndex];
            arr[negIndex] = temp;
            negIndex++;
        }
    }
}
```

Q4. Question: Given two arrays of equal size n and an integer k. The task is to permute both arrays such that the sum of their corresponding element is greater than or equal to k.

Answer in Java:

java

Copy code

```
public static String canPermuteToSumK(int[] a, int[] b, int k) {
    int n = a.length;

    Arrays.sort(a);
    Arrays.sort(b);

    int i = 0, j = n - 1;

    while (i < n && j >= 0) {
        int sum = a[i] + b[j];
        if (sum < k) {
            return "No";
        }
        i++;
        j--;
    }
```

```
    return "Yes";
}
```

Q5. Question: An interval is represented as a combination of start time and end time. Given a set of intervals, check if any two intervals intersect.

Answer in Java:

```
java
Copy code
public static boolean hasIntersectingIntervals(int[][] intervals) {
    int n = intervals.length;

    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (intervals[i][1] >= intervals[j][0] && intervals[i][0] <= intervals[j][1]) {
                return true; // Intervals i and j intersect
            }
        }
    }

    return false; // No intersecting intervals found
}
```