Ans7) public class InversionCount {

// Function to count inversions

public static int countInversions(int[] arr) {

return mergeSort(arr, 0, arr.length - 1);

}

// Recursive merge sort with inversion counting

private static int mergeSort(int[] arr, int left, int right) {

int count = 0;

if (left < right) {

int mid = (left + right) / 2;

count += mergeSort(arr, left, mid); // Left subarray

count += mergeSort(arr, mid + 1, right); // Right subarray

count += merge(arr, left, mid, right); // Merge step

}

return count;

}

// Merge two sorted halves and count inversions

private static int merge(int[] arr, int left, int mid, int right) {

int n1 = mid - left + 1;

int n2 = right - mid;

int[] L = new int[n1];

int[] R = new int[n2];

for (int i = 0; i < n1; i++) L[i] = arr[left + i];

for (int j = 0; j < n2; j++) R[j] = arr[mid + 1 + j];

int i = 0, j = 0, k = left, swaps = 0;

while (i < n1 && j < n2) {

if (L[i] <= R[j]) {

arr[k++] = L[i++];

} else {

arr[k++] = R[j++];

swaps += (n1 - i); // All remaining L[i..n1-1] are > R[j]

}

}

while (i < n1) arr[k++] = L[i++];

while (j < n2) arr[k++] = R[j++];

return swaps;

}

// Driver Code

public static void main(String[] args) {

int[] arr = {2, 4, 1, 3, 5};

int result = countInversions(arr);

System.out.println("Number of inversions: " + result);

}

}