UNITED INTERNATIONAL UNIVERSITY
Department of Computer Science and Engineering (CSE)

Course Title: Data Structure & Algorithm Lab II **Course Code: CSE2218**

ASSIGNMENT 01: Divide and Conquer

Q1: Maximum Subarray Implementation

Given an integer array `nums`, find the contiguous subarray (containing at least one number) which has the largest sum and return *its* ***sum***.

A **subarray** is a **contiguous** part of an array.

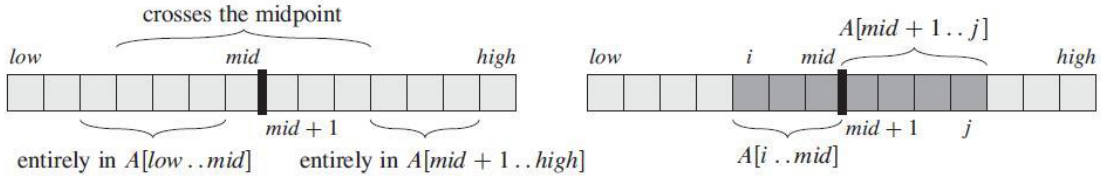
Example:

```
Input: nums = [-2, 1,-3, 4,-1, 2, 1,-5, 4]
Output: 6
Explanation: [4,-1, 2, 1] has the largest sum = 6.
```

Steps to follow:

Each time divide the array into two halves.

a. Recursive call the first half to return you the maximum subarray sum of that portion.
b. Recursive call the second half to return you the maximum subarray sum of that portion.
c. Maximum subarray may exist around the mid-point. So, calculate the maximum subarray sum across the split boundary.



How to calculate the maximum crossing subarray sum:

```
FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
1  left-sum = −∞
2  sum = 0
3  for i = mid downto low
4      sum = sum + A[i]
5      if sum > left-sum
6          left-sum = sum
7          max-left = i
8  right-sum = −∞
9  sum = 0
10 for j = mid + 1 to high
11     sum = sum + A[j]
12     if sum > right-sum
13         right-sum = sum
14         max-right = j
15 return (max-left, max-right, left-sum + right-sum)
```

ii. Base condition: If the array contains only 1 item then the maximum subarray sum is the item itself.

Q2: Count Inversions

Given a set of integers, find out the total number of inversions.

Input:

- The first line contains **n**, the number of integers.
- The second line contains n space separated integers $\mathbf{a_0 \ldots a_{n-1}}$

Output: The output is an integer indicating the total number of violations.

Example:

Sample Input	Sample Output	Explanation
5 4 5 6 7 1	4	1 violates which requires 4 violation to be fixed as 1 4 5 6 7.
5 5 4 3 2 1	10	4 violates which requires 1 violation to be fixed as 4 5 3 2 1. Then again 3 violates which requires 2 violation to be fixed as 3 4 5 2 1. Then again 2 violates which requires 3 violation to be fixed as 2 3 4 5 1. Then again 1 violates which requires 4 violation to be fixed as 1 2 3 4 5. Total violations required = 1+2+3+4 =10

You can modify the **mergesort** algorithm in the following way to solve this problem:

1. Divide the array into two equal/almost equal halves in each of the recursive step until the base case is reached.
2. Create a merge function that counts the number of inversions when two halves of the array are merged in the following way:
    a. Let us consider i be the index of the first half and j be the index of the second half.
    b. If a[i] > a[j[then there are (mid – i) inversions because the left and right subarrays are sorted, so all the remaining elements in the left subarray (a[i+1], a[i+2], a[i+3]) will be greater than a[i]
3. Create the required recursive function that divides the array into right and left halves and find the answer by summing the number of inversions in the first half and the number of inversions in the second half by simple summation.
    a. The base case is when there is only one element in the given half
4. Print the final answer.

Q3: Quick Sort Implementation

Given an integer array `nums`. Sort the array in descending order using Quicksort algorithm and print the array.

Sample Input	Sample Output
6 4 5 6 7 1 3	7 6 5 4 3 1

Q4: Shuffle Storm

Mr. Habib got an interesting gift on his birthday that is an array of **N distinct** integers which are
sorted in **descending order**. One evening, Habib was playing with the array and suddenly a storm named "Shuffle Storm" came and shuffled the array. Now, Habib is feeling very sad. We don't want to see anyone sad. Right? So, we need to restore the array to its initial position.
For that, for each index of the array, you should figure out how many moves it needed and in which direction to get its initial position.
Let's make Habib happy.

The first line of the input represents the number of elements in the array and the next line has **N** distinct integers mean the shuffled array.
note: If no move needed for any index, print "0" in a single line.
See example for clarification.

Sample Input	Sample Output
8 3 5 6 9 4 1 2 7	5 Right 2 Right 0 3 Left 0 2 Right 0 6 Left

Note: Overall Time Complexity must be N*long(N)
Hint: You can find the index of an element using Binary Search in a sorted array.