3.  Given an array of positive integers. Write a C Program to find inversion count of array.

**Inversion Count:** For an array, inversion count indicates how far (or close) the array is from being sorted. If array is already sorted then inversion count is 0. If array is sorted in reverse order that inversion count is the maximum.  
Formally, two elements a[i] and a[j] form an inversion if a[i] > a[j] and i < j.  
**Input:**  
The first line of input contains an integer T denoting the number of test cases. The first line of each test case is N, the size of array. The second line of each test case contains N elements.  
**Output:**  
Print the inversion count of array.  
**Constraints:**  
1 ≤ T ≤ 100  
1 ≤ N ≤ 107  
1 ≤ C ≤ 1018  
**Example:  
Input:**  
1  
5  
2 4 1 3 5  
**Output:**  
3  
**Explanation:  
Testcase 1:** The sequence 2, 4, 1, 3, 5 has three inversions (2, 1), (4, 1), (4, 3)

#include <stdio.h>

int getInvCount(int arr[], int n)

{

int i,j;

int inv\_count = 0;

for ( i = 0; i < n - 1;i++)

for ( j = i + 1; j < n; j++)

if (arr[i] > arr[j])

inv\_count++;

return inv\_count;

}

int main(int argv, char\*\* args)

{

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

int n = sizeof(arr) / sizeof(arr[0]);

printf(" Number of inversions are %d \n", getInvCount(arr, n));

return 0;

}

**Output:**

