1. Given two sorted arrays a1 and a2 of size m and n. Find the median of the two sorted arrays in O(log (m + n)).

Example:

[1, 4, 5, 8], [2, 3, 6] -> 4

[1, 4, 5], [2, 3, 6] -> 3.5

Input: 2 arrays

Output: double

C++: double FindMedian(const std::vector<int>& a1, const std::vector<int>& a2)  
Java: class Main { public static double findMedian(List<Integer> a1, List<Integer> a2) }  
Python: def find\_median(lst1, lst2)

1. Given an array with each row and each column sorted in ascending order, return the kth largest element.

Example:

[[ 1, 3, 5, 10],

[ 6, 11, 15, 20],

[ 8, 13, 17, 23]], 5 -> 13

[[ 1, 8],

[ 13, 19],

[ 20, 23]], 2 -> 20

Input: 2d array, int

Output: int

C++: int KthLargest(const vector<vector<int>>& matrix, int k)  
Java: class Main { public static int kthLargest(List<List<Integer>> matrix, Integer k) }  
Python: def kth\_largest(matrix, k)

1. Given an array sorted in non-decreasing order, return a new array containing squares of each number in non-decreasing order.

Example:

[-3, -2, 0, 4, 6] -> [0, 4, 9, 16, 36]

[-5, -3, 2, 3, 10] -> [4, 9, 9, 25, 100]

Input: array

Output: array

C++: std::vector<int> SortSquares(const std::vector<int>& a)  
Java: class Main { public static List<Integer> sortSquares(List<Integer> a) }  
Python: def sort\_squares(lst)

1. Given an array of objects colored red, green, and blue. Sort them in place so that the same color objects are adjacent. Order the colors red, green, blue represented by integers 0, 1, 2 respectively. Try to come up with a one-pass algorithm that uses constant space.

Example:

[2, 1, 1, 0, 2, 1] -> [0, 1, 1, 1, 2, 2]

[1, 1, 2, 0, 2] -> [0, 1, 1, 2, 2]

Input: array

Output: array

C++: void SortColors(std::vector<int>& a)  
Java: class Main { public static void sortColors(List<Integer> a) }  
Python: def sort\_colors(lst)