// 4-Median\_of\_Two\_Sorted\_Arrays.cpp //

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

class Solution {

public:

double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {

int m = nums1.size();

int n = nums2.size();

if (m > n) { // to ensure m<=n

vector<int> temp = nums1; nums1 = nums2; nums2 = temp;

int tmp = m; m = n; n = tmp;

}

int iMin = 0, iMax = m, halfLen = (m + n + 1) / 2;

while (iMin <= iMax) {

int i = (iMin + iMax) / 2;

int j = halfLen - i;

if (i < iMax && nums2[j - 1] > nums1[i]) {

iMin = i + 1; // i is too small

}

else if (i > iMin && nums1[i - 1] > nums2[j]) {

iMax = i - 1; // i is too big

}

else { // i is perfect

int maxLeft = 0;

if (i == 0) { maxLeft = nums2[j - 1]; }

else if (j == 0) { maxLeft = nums1[i - 1]; }

else { maxLeft = max(nums1[i - 1], nums2[j - 1]); }

if ((m + n) % 2 == 1) { return maxLeft; }

int minRight = 0;

if (i == m) { minRight = nums2[j]; }

else if (j == n) { minRight = nums1[i]; }

else { minRight = min(nums2[j], nums1[i]); }

return ((double)maxLeft + (double)minRight) / 2.0;

}

}

return 0.0;

}

};

int main()

{

Solution sol;

vector<int> num1 = { 1, 3 };

vector<int> num2 = { 2 };

std::cout << sol.findMedianSortedArrays(num1, num2) << "\n"; // 2

vector<int> num3 = { 1, 2 };

vector<int> num4 = { 3, 4 };

std::cout << sol.findMedianSortedArrays(num3, num4) << "\n"; // 2.5

}