**Union of arrays with duplicate elements**

class Solution {

public static int findUnion(int a[], int b[]) {

Arrays.sort(a);

Arrays.sort(b);

int ptr1 = 0;

int ptr2 = 0;

int m = a.length;

int n = b.length;

int count = 0;

int last = Integer.MIN\_VALUE;

while (ptr1 < m && ptr2 < n){

if (a[ptr1] < b[ptr2]){

if (a[ptr1] != last){

count++;

last = a[ptr1];

};

ptr1++;

}else if (b[ptr2] < a[ptr1]){

if (b[ptr2] != last){

count++;

last = b[ptr2];

};

ptr2++;

}else{

if (last != a[ptr1]){

count++;

last = a[ptr1];

};

ptr1++;

ptr2++;

}

}

while (ptr1 < m){

if (a[ptr1] != last){

count++;

last = a[ptr1];

}

ptr1++;

}

while (ptr2< n){

if (b[ptr2] != last){

count++;

last = b[ptr2];

}

ptr2++;

}

return count;

}

}

**TC: O(N)**

**SC: O(1)**

**Equilibrium Point**

class Solution {

// Function to find equilibrium point in the array.

public static int equilibriumPoint(int arr[]) {

int n = arr.length;

int[] psum = new int[n];

int[] ssum = new int[n];

int fsum = 0, lsum = 0;

for (int i = 0; i<n; i++){

fsum+=arr[i];

lsum+=arr[n-i-1];

psum[i] = fsum;

ssum[n-i-1] = lsum;

}

for (int i = 0; i<n; i++){

if (psum[i] == ssum[i]) return i+1;

}

return -1;

}

}

**TC: O(2N)**

**SC: O(2N)**