Problem 2

a.

Algorithm Merge(arr1,arr2)

Input: arr1 array of numbers

Input: arr2 array of numbers

Output: k array containing arr1 and arr2 inputs

sumLength ← arr1.length + arr2.length **//sum of the length of arr1 and arr2**

k ← new Integer[sumLength] **//create a new array k of length equal to the sum of the length arr1 and arr2**

c ←0

for i ←0 to k.length – 1 **do**

if i <= arr1.length-1 then

    k[i] ← arr1[i]

    { increment counter c}

{ increment counter i }

for j ←c , x←0 to k.length – 1 **do**

k[j] ← arr2[x]

for m ← 0 to k.length **do**

for n ← m + 1 to k.length **do**

    if k[m] > k[n] then

        temp ← k[m]

        k[m] ← k[n]

        k[n] ← temp

return k

b.

Algorithm Merge(arr1,arr2)	--->	Operation
sumLength ← arr1.length + arr2.length	→	**3**
k ← new Integer[sumLength]	→	2
c ←0	→	1
for i ←0 to k.length – 1 **do**	→	**1 + n**
if i <= arr1.length-1 then	→	2(n-1)
k[i] ← arr1[i]	→	3(n-1)
{ increment counter c}	→	2(n-1)
{ increment counter i }	→	2(n-1)
for j ←c , x←0 to k.length – 1 **do**	→	**1 + n**
k[j] ← arr2[x]	→	3(n-1)
for m ← 0 to k.length **do**	→	**n + 1**
for n ← m + 1 to k.length **do**	→	**n(n + 1)**
if k[m] > k[n] then	→	3(n(n-1))
temp ← k[m]	→	2(n(n-1))
k[m] ← k[n]	→	3(n(n-1))
k[n] ← temp	→	2(n(n-1))
return k	→	1
Total		**$11n^2$**

Since $11n^2$ is $0(n^2)$, algorithm runs in $0(n^2)$ time.

c.

```
int[] merge(int[] arr1,int[] arr2){
    int sumLength = arr1.length+arr2.length;
    int[] k = new int[sumLength];

    int c = 0;
    for (int i = 0; i < k.length; i++) {
        if(i <= arr1.length-1) {
            k[i] = arr1[i];
            c++;
        }
    }

    for (int j = c,x = 0; j < k.length; j++,x++){
            k[j] = arr2[x];

    }

    for (int m = 0; m < k.length; m++){
        for (int n = m + 1; n < k.length; n++) {
            if (k[m] > k[n]) {
                int temp = k[m];
                k[m] = k[n];
                k[n] = temp;
            }
        }
    }

    return k;
}
```