Lab03

Question2:

```
static void sort(int[] arr){
    int len = arr.length;
    Boolean alreadySorted = false;
    for(int i = 0; i < len && !alreadySorted; ++i) {
        alreadySorted = true;
        for(int j = 0; j< len -1; ++j) {
            if(arr[j] > arr[j+1]){
                alreadySorted = false;
                swap(arr, j,j+1);
            }
        }
    }
}
```

If array is already sorted then alreadySorted variable will be true and outer loop won't execute after first execution. So running time will be

Outer loop = O(1) since alreadySorted variable will be true after first execution of inner loop

Inner loop = O(n) since outer loop will be executed once.

Hence best case running time for this enhancement will be O(n)

Question3:

```
static void sort(int[] arr){
    int len = arr.length;
    Boolean alreadySorted = false;
    for(int i = 0; i < len && !alreadySorted; ++i) {
        alreadySorted = true;
        for(int j = 0; j< len-i-1; ++j) {
            if(arr[j] > arr[j+1]){
                alreadySorted = false;
                swap(arr, j,j+1);
            }
        }
    }
}
```

By altering inner loop to end at len-i-1 each time, it will decrease number of statements executed each time i increase. So let's assume array length = n. The inner loop will be executed like the following

i	n-i-1	
1	n-2	
2	n-3	
3	n-4	
4	n-5	
5	n-6	
n-2	1	
n-1	0	

So total execution time will be n-2 + n-3 + n-4 + ….. + 2 + 1 + 0

Since n is the most significant term. This enhancement will increase algorithm speed to double of its old speed. Running time is still O(n) though.

Question4:

```
package algorithm.lab03Q3;

public class Lab03Que03{

	private  int[] finalSortedArray;
	public  int[] sort(int[] arr){
		if (arr == null || arr.length == 0)
			return null;
		finalSortedArray = new int[arr.length];
		int zeros = 0, ones = 0, twos = 0, temp = 0;
		for (int i = 0 ; i < arr.length; i ++){
			temp = arr[i];
			if (temp == 0)
				zeros++;
			else if (temp == 1)
				ones ++;
			else
				twos ++;
		}
		print(0, zeros, finalSortedArray, 0);
		print(1, ones, finalSortedArray, zeros);
		print(2, twos, finalSortedArray, zeros+ones);
		return finalSortedArray;
	}

	private  void print(int number, int count, int[] arr, int startIndex){
		for (int i = 0; i < count; i++){
			arr[startIndex++] = number;
		}
	}
	public static void main(String[] args) {

		Lab03Que03 LQ = new Lab03Que03();
		int[] arr={0,0,1,2,2,2,1,0};
		int[]  finalSortedArray = LQ.sort(arr);
		for(int i =0; i< finalSortedArray.length; i++)
			System.out.print(finalSortedArray[i] + " ");

	}
}
```