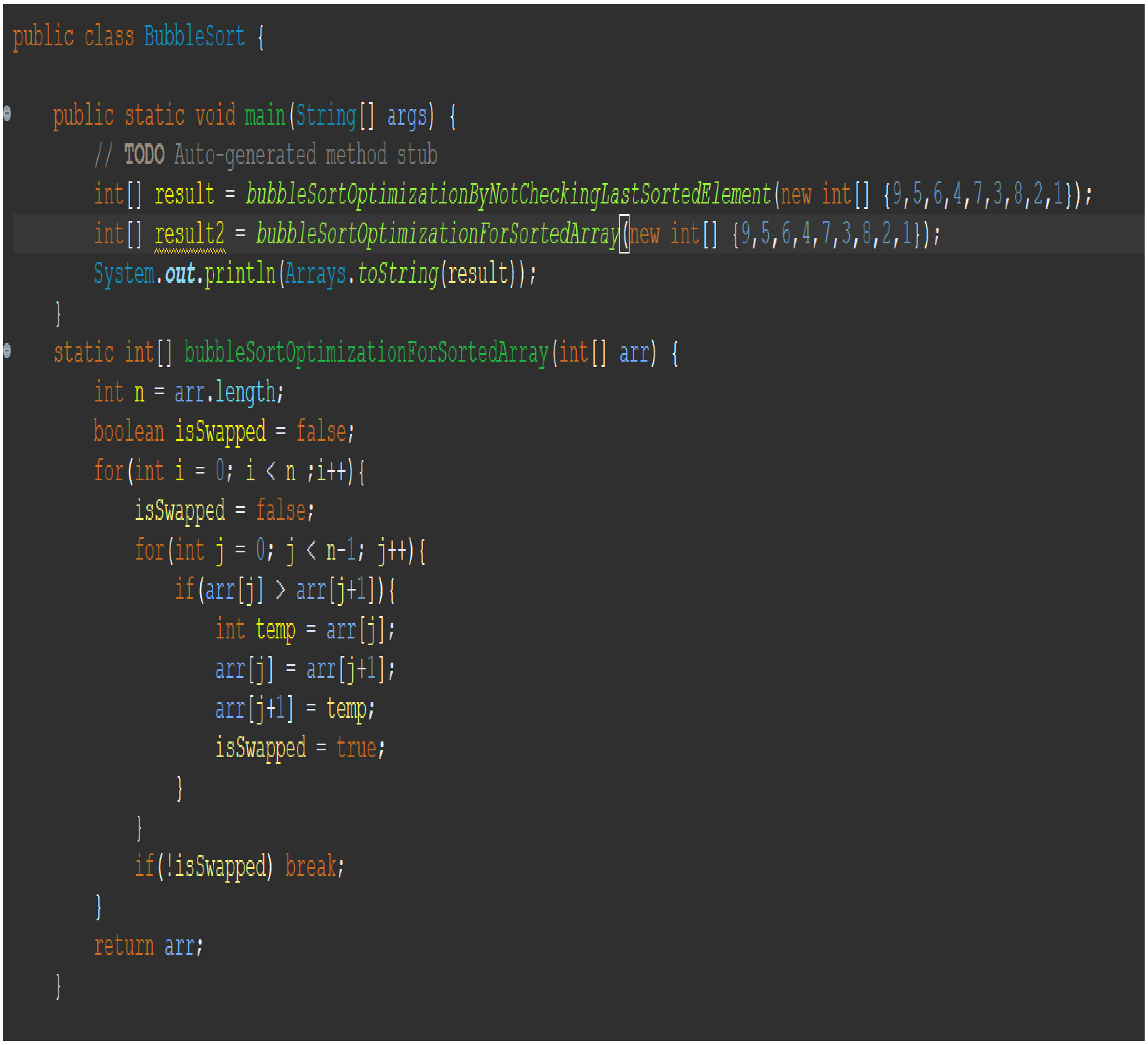
Lab 3

Student ID	Name
614760	Omkar Nath Chaudhary
614732	Sushil Subedi
614604	Rahul Niraula
614600	Shrawan Adhikari

Q2.Write JavaCode solution to improve bubble sort for the scenario in which the input is already sorted.



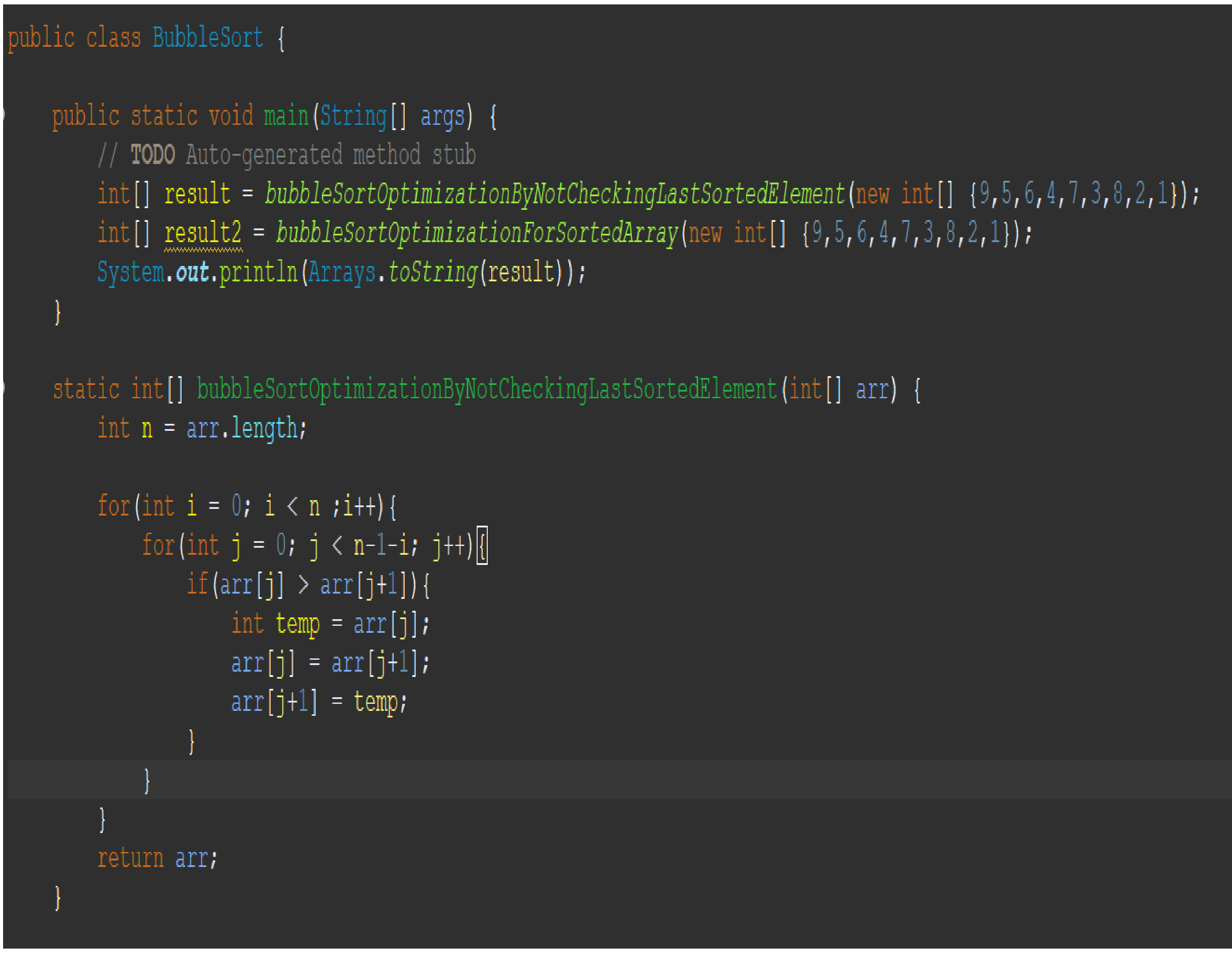
```java
public class BubbleSort {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int[] result = bubbleSortOptimizationByNotCheckingLastSortedElement(new int[] {9,5,6,4,7,3,8,2,1});
        int[] result2 = bubbleSortOptimizationForSortedArray(new int[] {9,5,6,4,7,3,8,2,1});
        System.out.println(Arrays.toString(result));
    }
    static int[] bubbleSortOptimizationForSortedArray(int[] arr) {
        int n = arr.length;
        boolean isSwapped = false;
        for(int i = 0; i < n ;i++){
            isSwapped = false;
            for(int j = 0; j < n-1; j++){
                if(arr[j] > arr[j+1]){
                    int temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                    isSwapped = true;
                }
            }
            if(!isSwapped) break;
        }
        return arr;
    }
```

Time Complexity: If the array is sorted then, it runs only the inner loop. Complexity is O(n).
Here, n is O(n), If f(n) <= cg(n)

n <= cn, for all c > 2, this is always true. So, n is O(n)

Therefore, the complexity is O(n)

Q3. Write JavaCode solution for improving Bubble sort by cutting the running time for every ith pass.



```
public class BubbleSort {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int[] result = bubbleSortOptimizationByNotCheckingLastSortedElement(new int[] {9,5,6,4,7,3,8,2,1});
        int[] result2 = bubbleSortOptimizationForSortedArray(new int[] {9,5,6,4,7,3,8,2,1});
        System.out.println(Arrays.toString(result));
    }

    static int[] bubbleSortOptimizationByNotCheckingLastSortedElement(int[] arr) {
        int n = arr.length;

        for(int i = 0; i < n ;i++){
            for(int j = 0; j < n-1-i; j++){
                if(arr[j] > arr[j+1]){
                    int temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
            }
        }
        return arr;
    }
```

Time Complexity: O(n)*O(n(n-1)/2) = O(n^2(n-1)/2);
Which is half of the Bubble Sort Time complexity.

Q4. Given array of n, integers that belong to the set {0,1, 2}. Write an sorting algorithm to sort the array and explain why it runs in O(n) time.

```java
public static void main(String[] args) {
   int[] inputArr = new int[]{2, 2, 1, 0, 0, 1, 2, 0, 1, 2, 1, 1};
   int[] sortedArr = sortSetOf3Integers(inputArr);
   System.out.println(Arrays.toString(sortedArr));
}

public static int[] sortSetOf3Integers(int[] arr) {
   int[] resultArr = new int[arr.length];
   int aCount = 0, bCount = 0, cCount = 0;
    // Counting numbers of 0's , 1 and 2
   for (int i = 0; i < arr.length; i++) {      // O(n)
       if (arr[i] == 0) aCount++;
       if (arr[i] == 1) bCount++;
       if (arr[i] == 2) cCount++;
   }
   // Creating sorted resultArr using numbers count (start-end)
   for (int i = 0; i < resultArr.length; i++) {     // O(n)
       if (i < aCount) {
           resultArr[i] = 0;
       } else if (i < (aCount+bCount)) {
           resultArr[i] = 1;
       } else {
           resultArr[i] = 2;
       }
   }
   return resultArr;
}
```

Complexity: O(n) + O(n) = 2*O(n) = **O(n)**

2n is O(n), If f(n) <= cg(n)

2n <= cn, for all c > 2, this is always true. So, 2n is O(n)

Therefore, the complexity is O(n)