3.1 In the bubbleSort.java program (Listing 3.1) and the BubbleSort Workshop applet, the in index always goes from left to right, finding the largest item and carrying it toward out on the right. Modify the bubbleSort() method so that it’s bidirectional. This means the in index will first carry the largest item from left to right as before, but when it reaches out, it will reverse and carry the smallest item from right to left. You’ll need two outer indexes, one on the right (the old out) and another on the left.

public void bubbleSort()

    {

        int out, in;

        for(out = nElems-1; out>1; out--)

            for(in = 0; in<out; in++)

                if(a[in]<a[in+1])

                    swap(in,in+1);

    }

3.2 Add a method called median() to the ArrayIns class in the insertSort.java program (Listing 3.3). This method should return the median value in the array. (Recall that in a group of numbers half are larger than the median and half are smaller.) Do it the easy way.

 public double median()

    {

        int n = nElems;

        if(n%2!=0)

            return a[n/2];

        return (double)(a[(n-1)/2]+a[n/2])/2;

    }

3.3 To the insertSort.java program (Listing 3.3), add a method called noDups() that removes duplicates from a previously sorted array without disrupting the order. (You can use the insertionSort() method to sort the data, or you can simply use main() to insert the data in sorted order.) One can imagine schemes in which all the items from the place where a duplicate was discovered to the end of the array would be shifted down one space every time a duplicate was discovered, but this would lead to slow O(N2 ) time, at least when there were a lot of duplicates. In your algorithm, make sure no item is moved more than once, no matter how many duplicates there are. This will give you an algorithm with O(N) time.

public void noDups(){

        int j = 0;

        for (int i=0; i < nElems-1; i++){

            if (a[i] != a[i+1]){

                a[j++] = a[i];

            }

        }

        a[j++] = a[nElems-1];

        nElems = j;

    }