```java
// SortTools.java
/*
 * EE422C Project 1 submission by
 * Replace <...> with your actual data.
 * Dhruv Sandesara
 * djs3967
 * 15495
 * Spring 2017
 * Slip days used: 0
 */

package assignment1;
public class SortTools {
    /**
      * This method tests to see if the given array is sorted.
      * @param x is the array
      * @param n is the size of the input to be checked
      * @return true if array is sorted
      */
    public static boolean isSorted(int[] x, int n) {

        if(n==0||x.length==0)
            return false;//checking additional pre conditions


        for(int i=0;i<n-1;i++){
            if (x[i] > x[i+1]) {//if in non decreasing order
            return false;//return false
            }

        }
        return true;
    }


    //Find function that returns index from a sorted array if element found. if not
     found return -1
    public static int find(int[] nums, int n, int v){

        int lowerBound=0,upperBound=n-1;
        int mid=0;

        while (lowerBound<=upperBound){
            mid=(lowerBound+upperBound)/2;

            if(nums[mid]==v)//number found
                return mid;
            else if(nums[mid]<v)//number might be in the part of the array with
             bigger elemnts
                lowerBound=mid+1;
            else//number might be in the part of the array with smaller elemnts
                upperBound=mid-1;

        }
```

```
        return -1;//not found

    }


//    Return a newly created array of integers with the following properties. o The
 contents of the new array include the first n elements of
//    nums and the value v.
//    o The contents of the new array are sorted in non-decreasing
//    order.
//    o If the first n elements of nums contain at least one copy of the
//    value v, then the new array will contain n values (i.e. do not add
//            another copy of v if it is already in nums).
//    o If the first n elements of nums do not contain v then the new
//    array will contain n+1
//    values (i.e. the original contents plus v).
    public static int[] insertGeneral(int[] nums, int n, int v){

        if(find(nums,n,v)!=-1)
            return nums;//this is the case when v is already in the array


        int arr[]= new int[n+1];

        int vAlreadyInserted=0;

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

            if(vAlreadyInserted==0&&v<nums[i]) {
                //v has not been inserted yet and v is supposed to be inserted at
                 this spot to maintain the assending order
                arr[i]=v;
                vAlreadyInserted=1;
            }
            else
                arr[i]=nums[i-vAlreadyInserted];//vAlreadyInserted keeps track of
                 which element position to put into the new array
        }


        return arr;

    }


    public static int insertInPlace(int[] nums, int n, int v) {


        if (find(nums, n, v) != -1)//we do this test again so that we make sure the
         nums array is not modified into size n by the insert general
            return n;//this is the case when v is already in the array
```

```
        nums= insertGeneral(nums, n, v);
        return n+1;

    }

//takes an array and sorts the first n elements
    public static void insertSort(int[] nums, int n){


        int temp=0,tempIndex=0,currentElementIndex=1;

        while(currentElementIndex<n){//i is the iterator and sorting ends when all
         elements are sorted

            if(nums[currentElementIndex]>nums[currentElementIndex-1]){
                currentElementIndex++;//the element is in ascending order so move
                 onto next case
            }
            else{//need to put current element in the proper position in the left
             side of array which is sorted
                temp=nums[currentElementIndex];//temp has the element to be inserted
                tempIndex=currentElementIndex-1;
                while((tempIndex>=0)&&(nums[tempIndex]>temp)){
                    nums[tempIndex+1]=nums[tempIndex];//keeps shifting the larger
                     elements to the right
                    tempIndex--;//decrement index so it can happen again
                }
                //now at tempindex we have an element smaller then the number we
                 want to insert so we shall insert it
                nums[tempIndex+1]=temp;

                currentElementIndex++;

            }
        }

        return;//sort completed

    }

}
```