

```

1 // SortTools.java
2 /*
3  * EE422C Project 1 submission by
4  * Replace <...> with your actual data.
5  * <Tao Zhu>
6  * <tz3694>
7  * <15455>
8  * Spring 2017
9  * Slip days used:
10 */
11
12 package assignment1;
13 public class SortTools {
14     /**
15      * This method tests to see if the given array is sorted.
16      * @param x is the array
17      * @param n is the size of the input to be checked
18      * @return true if array is sorted
19      */
20     public static boolean isSorted(int[] x, int n)
21     {
22         if(n == 0 || x.length == 0)
23         {
24             return false;
25         }
26         for(int i = 0; i < n-1; i++)
27         {
28             if(x[i] > x[i+1])
29             {
30                 return false;
31             }
32         }
33         return true;
34     }
35
36     /**
37      * This method find whether a value exists in an array.
38      * @param nums is the array
39      * @param n is the size of the input to be checked
40      * @param v is the value to be checked
41      * @return the position of the value. Return -1 if v is not found.
42      */
43     public static int find(int[] nums, int n, int v)
44     {
45         int low = 0, high = n - 1, mid=0;
46         while(low <= high)
47         {
48             mid = (low + high) / 2;
49             if(nums[mid] == v)
50             {
51                 return mid;
52             }
53             else if(nums[mid] < v)
54             {
55                 low = mid + 1;
56             }
57             else
58             {
59                 high = mid - 1;
60             }
61         }
62         return -1;
63     }
64
65     /**
66      * This method insert a value to a sorted array.

```

```

67      * @param nums is the array
68      * @param n is the size of the input to be checked
69      * @param v is the value to be inserted
70      * @return the new array after insertion
71      */
72      public static int[] insertGeneral(int[] nums, int n, int v)
73      {
74          int pos = find(nums, n, v);
75          int[] new_nums;
76          if(pos != -1)
77          {
78              new_nums = new int[n];
79              for(int i = 0; i < n; i++)
80              {
81                  new_nums[i] = nums[i];
82              }
83          }
84          else
85          {
86              new_nums = new int[n+1];
87              int insert_pos = 0;
88              for(insert_pos = 0; insert_pos < n + 1; insert_pos++)
89              {
90                  if(insert_pos < n && nums[insert_pos] < v) //edge case: insert_pos is
91                      at the end of the array
92                  {
93                      new_nums[insert_pos] = nums[insert_pos];
94                  }
95                  else
96                  {
97                      new_nums[insert_pos] = v;
98                      break;
99                  }
100              }
101              for(int i = insert_pos + 1; i < n + 1; i++)
102              {
103                  new_nums[i] = nums[i-1];
104              }
105          }
106          return new_nums;
107      }
108  }
109
110  /**
111   * This method insert a value in a sorted array in place.
112   * @param num is the array
113   * @param n is the size of the input to be checked
114   * #param v is the value to be inserted
115   * @return the length of the new array
116   */
117  public static int insertInPlace(int[] nums, int n, int v)
118  {
119      int pos = find(nums, n, v);
120      if(pos != -1)
121      {
122          return n;
123      }
124      else
125      {
126          int insert_pos = 0;
127          for(insert_pos = 0; insert_pos < n + 1; insert_pos++)
128          {
129              if(insert_pos < n && nums[insert_pos] < v) //edge case: insert_pos is
130                  at the end of the array
131              {

```

```

131         //Do nothing
132     }
133     else
134     {
135         break;
136     }
137 }
138
139 for(int i = n; i > insert_pos; i--) //precondition: n < nums.length
140 {
141     nums[i] = nums[i-1];
142 }
143 nums[insert_pos] = v;
144 return (n + 1);
145 }
146 }
147
148 /**
149  * This method sorts an array.
150  * @param nums is the array
151  * @param n is the size of the input to be sorted
152  */
153 public static void insertSort(int[] nums, int n)
154 {
155     for(int i = 1; i < n; i++)
156     {
157         int key = nums[i];
158         for(int j = i - 1; j >= 0; j--)
159         {
160             if(nums[j] > key)
161             {
162                 nums[j + 1] = nums[j];
163             }
164             else
165             {
166                 nums[j + 1] = key;
167                 break;
168             }
169         }
170     }
171 }
172 }
173

```