MAC 286

Please submit your work as .java file on the blackboard.

To the ArrayIns class in the insertSort.java program (Listing 3.3), add the following methods:

1. oddEvenSort() to sort the array elements. The idea is to repeatedly make two passes through the array. On the first pass you look at all the pairs of items, a[j] and a[j+1], where j is odd (j = 1, 3, 5, …). If their key values are out of order, you swap them. On the second pass you do the same for all the even values (j = 2, 4, 6, …). You do these two passes repeatedly until the array is sorted. Replace the insertionSort () in the main method with an oddEvenSort() method. Make sure it works for varying amounts of data. You'll need to figure out how many times to do the two passes.

2. median() method to returns the median value in the array. (Recall that in a group of numbers half are larger than the median and half are smaller.).

Note: you can download the insertSort.java program code from the blackboard. (I also included the class below)

Sample output of program once the updates:

```
Array Before sorting
77 99 44 55 22 88 11 0 66 33
Array After sorting using oddEvenSort() method
0 11 22 33 44 55 66 77 88 99
Median is 55
Array after insert  109 and 85
0 11 22 33 44 55 66 77 85 88 99 109
Median after insert 109 and 85 is 66
```

```
// insertSort.java
// demonstrates insertion sort
// to run this program: C>java InsertSortApp
//--------------------------------------------------------------
```

```
class ArrayIns
   {
   private long[] a;                 // ref to array a
   private int nElems;               // number of data items
//--------------------------------------------------------------
   public ArrayIns(int max)          // constructor
      {
      a = new long[max];                 // create the array
      nElems = 0;                        // no items yet
      }
//--------------------------------------------------------------
   public void insert(long value)    // put element into array
      {
      a[nElems] = value;             // insert it
      nElems++;                      // increment size
      }
//--------------------------------------------------------------
   public void display()             // displays array contents
      {
      for(int j=0; j<nElems; j++)       // for each element,
         System.out.print(a[j] + " ");  // display it
      System.out.println("");
      }
//--------------------------------------------------------------
   public void insertionSort()
      {
      int in, out;

      for(out=1; out<nElems; out++)     // out is dividing line
         {
         long temp = a[out];            // remove marked item
         in = out;                      // start shifts at out
         while(in>0 && a[in-1] >= temp) // until one is smaller,
            {
            a[in] = a[in-1];            // shift item to right
            --in;                       // go left one position
            }
         a[in] = temp;                  // insert marked item
         }  // end for
      }  // end insertionSort()
//--------------------------------------------------------------
   }  // end class ArrayIns
////////////////////////////////////////////////////////////////
class InsertSortApp
   {
   public static void main(String[] args)
      {
      int maxSize = 100;            // array size
      ArrayIns arr;                 // reference to array
      arr = new ArrayIns(maxSize);  // create the array

      arr.insert(77);               // insert 10 items
      arr.insert(99);
      arr.insert(44);
```

```
      arr.insert(55);
      arr.insert(22);
      arr.insert(88);
      arr.insert(11);
      arr.insert(00);
      arr.insert(66);
      arr.insert(33);

      arr.display();                // display items

      arr.insertionSort();          // insertion-sort them

      arr.display();                // display them again
      }  // end main()
   }  // end class InsertSortApp
```