```
// takes the array of integer elements and sorts it in increasing order
void mergesort(int num[])
     mergesort(num, 0, num.length-1); // beginning index and last index to sort

// recursive
void mergesort(int num[], int low, int high)

     int mid
     if (low < high) // have subarray to sort (until single unit - index same)
          mid = (low + high) / 2          // midpoint of the subarray to sort
          mergesort(num, low, mid)        // recursively sort low to mid subarray
          mergesort(num, mid+1, high)   // recursively sort mid+1 to high subarray
          merge(num,low,mid,mid+1,high);  // merge two sorted subarrays

// merges two subarrays in sorted order
void merge(int[] a, int l1, int u1, int l2, int u2)

     // array 1: l1...u1 and array 2: l2 .. u2

     declare array temp of size of input array a

     // i is current index of left subarray; j of right subarray; k of temp array
     int i, j, k
     i=l1, j=l2, k=l1

     // loop as long as have elements in both left AND right subarray
      loop as long as i <= u1 && j <= u2

         if (a[i] <= a[j])   // left subarray el val <= right subarray el value
             temp[k] = a[i]   // copy left element to temp
             i=i+1     // increment to next left element
         else
             temp[k] = a[j]   // copy right element to temp
             j=j+1 // increment to next right element
          k=k+1

      // copy remaining elements in left subarray
      while (i<=u1)
           temp[k] = a[i]
           k=k+1  // increment index for temp array
           i=i+1  // increment index for left subarray

      // copy remaining elements in right subarray
      while (j<=u2)
           temp[k] = a[j]
           k=k+1  // increment index for temp array
           i=i+1  // increment index for right subarray

     // copy temp[] back into a[]
     int h = l1
      loop as long as h <= u2   // more elements to copy
           a[h] = temp[h]
           h=h+1
```