static void Mergesort(int arr[], int beg, int end) {

System.out.println("--------------------");

System.out.println("Mergesort(arr," + beg + "," + end + ")");

if (beg < end) {

// Find the middle point

int mid = beg + (end - beg) / 2;

System.out.println("MID=" + mid);

// Sort first and second halves

Mergesort(arr, beg, mid);

Mergesort(arr, mid + 1, end);

// Merge the sorted halves

merge(arr, beg, mid, end);

}

}

static void merge(int arr[], int beg, int mid, int end) {

System.out.println("--------------------");

System.out.println("--------------------");

System.out.println("Merge(arr," + beg + "," + mid + "," + end + ")");

// Find sizes of two subarrays to be merged

int n1 = mid - beg + 1;

int n2 = end - mid;

// Create temp arrays

int LeftArray[] = new int[n1];

int RightArray[] = new int[n2];

// Copy data to temp arrays

for (int i = 0; i < n1; ++i) {

LeftArray[i] = arr[beg + i];

}

for (int j = 0; j < n2; ++j) {

RightArray[j] = arr[mid + 1 + j];

}

// printarr(LeftArray);

// printarr(RightArray);

// Merge the temp arrays

// Initial indices of first and second subarrays

int i = 0, j = 0;

// Initial index of merged subarray array

int k = beg;

while (i < n1 && j < n2) {

if (LeftArray[i] <= RightArray[j]) {

arr[k] = LeftArray[i];

i++;

} else {

arr[k] = RightArray[j];

j++;

}

k++;

}

// Copy remaining elements of LeftArray[] if any

while (i < n1) {

arr[k] = LeftArray[i];

i++;

k++;

}

// Copy remaining elements of RightArray[] if any

while (j < n2) {

arr[k] = RightArray[j];

j++;

k++;

}

printarr(arr);

}

static void printarr(int a[]) {

System.out.println("");

for (int i = 0; i < a.length; i++) {

System.out.print(i + "\t");

}

System.out.println("");

for (int element : a) {

System.out.print(element + "\t");

}

System.out.println("");

}

public static void main(String[] args) {

int[] a = {11, 12, 15, 17, 19, 110, 111, 112};//{12, 15, 11, 17, 19, 112, 111, 110};

printarr(a);

Mergesort(a, 0, a.length - 1);

printarr(a);

}