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);

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

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

printarr(a);

}

static void swap(int[] arr, int i, int j) {

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

static int partition(int[] arr, int low, int high) {

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

System.out.println("partition(arr," + low + "," + high + ")");

// Choosing the pivot

int pivot = arr[high];

System.out.println("Pivot:" + pivot);

// Index of smaller element and indicates the right position of pivot found so far

int i = (low - 1);

for (int j = low; j <= high - 1; j++) {

// If current element is smaller than the pivot

if (arr[j] < pivot) {

// Increment index of smaller element

i++;

swap(arr, i, j);

}

}

swap(arr, i + 1, high);

printarr(arr);

return (i + 1);

}

static void quickSort(int[] arr, int low, int high) {

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

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

System.out.println("Quicksort(arr," + low + "," + high + ")");

if (low < high) {

// pi is partitioning index, arr[p] is now at right place

int pi = partition(arr, low, high);

System.out.println("Partition:" + pi);

// Separately sort elements before partition and after partition

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}