public void sort(int[] arr,int l,int r){

if (l < r) {

// Find the middle point

int m =l+ (r-l)/2;

// Sort first and second halves

sort(arr, l, m);

sort(arr, m + 1, r);

// Merge the sorted halves

merge(arr, l, m, r);

}

}

public static void main(String[] args) {

mergesort so =new mergesort();

int arr[] = { 12, 11, 13, 5, 6, 7 };

System.out.println("Before sorting the array");

so.printArray(arr,arr.length);

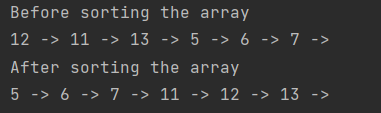
so.sort(arr,0,arr.length -1);

System.out.println("\nAfter sorting the array");

so.printArray(arr,arr.length);

}

**OUTPUT -**



public void merge(int[] arr,int l,int m,int r){

int n1 = m - l + 1;

int n2 = r - m;

/\* Create temp arrays \*/

int L[] = new int[n1];

int R[] = new int[n2];

/\*Copy data to temp arrays

\* \*/

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

L[i] = arr[l + i];

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

R[j] = arr[m + 1 + j];

// Initial indexes of first and second subarrays

int i = 0, j = 0 , k = l;

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

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

} else {

arr[k] = R[j];

j++;

} k++;

}

/\* Copy remaining elements if any \*/

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = R[j];

j++;

k++;

}

}