```c
//
// Created by hfwei on 2023/11/15.
//

#include <stdio.h>

#define LEN 7

/**
 * @brief Sort numbs[left .. right] using merge sort.
 * @param nums
 * @param left
 * @param right
 */
void MergeSort(int nums[], int left, int right);

/**
 * @brief Merge nums[left .. mid] and nums[mid + 1 .. right]
 * @param nums
 * @param left
 * @param mid
 * @param right
 */
void Merge(int nums[], int left, int mid, int right);

/**
 * @brief Copy src[left .. right] to dest[left .. right]
 * @param src
 * @param dest
 * @param left
 * @param right
 */
void Copy(const int src[], int dest[], int left, int right);

int main() {
  int numbers[LEN] = {38, 27, 43, 3, 9, 82, 10};
  MergeSort(numbers, 0, LEN - 1);

  for (int i = 0; i < LEN; i++) {
    printf("%d ", numbers[i]);
  }

  return 0;
}

void MergeSort(int nums[], int left, int right) {
  if (left == right) {
    return;
  }

  int mid = (left + right) / 2;
  MergeSort(nums, left, mid);      // Call the Mirror
  MergeSort(nums, mid + 1, right);  // Call the Mirror

  Merge(nums, left, mid, right);
}

void Merge(int nums[], int left, int mid, int right) {
  static int copy[LEN] = {0};

  int left_index = left;
  int right_index = mid + 1;
  int copy_index = left;

  while (left_index <= mid && right_index <= right) {
    if (nums[left_index] <= nums[right_index]) {
      copy[copy_index] = nums[left_index];
      left_index++;
    } else {
      copy[copy_index] = nums[right_index];
      right_index++;
    }

    copy_index++;
  }

  while (left_index <= mid) {
    copy[copy_index] = nums[left_index];
    left_index++;
    copy_index++;
  }

  while (right_index <= right) {
    copy[copy_index] = nums[right_index];
    right_index++;
    copy_index++;
  }

  Copy(copy, nums, left, right);
}

void Copy(const int src[], int dest[], int left, int right) {
  for (int i = left; i <= right; ++i) {
    dest[i] = src[i];
  }
}
```