15 merges sort

#include <stdio.h>

void merge(int arr[], int low, int mid, int high) {

int n1 = mid - low + 1;

int n2 = high - mid;

// Create temporary arrays

int left[n1], right[n2];

// Copy data to temporary arrays

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

left[i] = arr[low + i];

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

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

// Merge the temporary arrays back into arr[low..high]

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

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

if (left[i] <= right[j]) {

arr[k] = left[i];

i++;

} else {

arr[k] = right[j];

j++;

}

k++;

}

// Copy the remaining elements of left[] if there are any

while (i < n1) {

arr[k] = left[i];

i++;

k++;

}

// Copy the remaining elements of right[] if there are any

while (j < n2) {

arr[k] = right[j];

j++;

k++;

}

}

void merge\_sort(int arr[], int low, int high) {

if (low < high) {

// Find the middle point

int mid = low + (high - low) / 2;

// Sort first and second halves

merge\_sort(arr, low, mid);

merge\_sort(arr, mid + 1, high);

// Merge the sorted halves

merge(arr, low, mid, high);

}

}

void print\_array(int arr[], int n) {

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

printf("%d ", arr[i]);

printf("\n");

}

int main() {

int arr[] = {6, 5, 3, 1, 8, 7, 2, 4};

int n = sizeof(arr) / sizeof(arr[0]);

printf("Unsorted array: ");

print\_array(arr, n);

merge\_sort(arr, 0, n - 1);

printf("Sorted array: ");

print\_array(arr, n);

return 0;

} 