#include <iostream>

using namespace std;

void swap(int &a, int &b)

{

int tmp = b;

b = a;

a = tmp;

}

//Selection Sort ( Sắp xếp chọn)

void selectionSort(int arr[], int n)

{

int min;

for(int i = 0; i < n - 1; i++)

{

min = i;

for(int j = i + 1; j < n; j++)

{

if(arr[j] < arr[min])

min = j;

}

if(min != i)

swap(arr[min], arr[i]);

}

}

//Head sort

void heapify(int arr[], int N, int i)

{

int largest = i;

int l = 2 \* i + 1;

int r = 2 \* i + 2;

if (l < N && arr[l] > arr[largest])

largest = l;

if (r < N && arr[r] > arr[largest])

largest = r;

if (largest != i) {

swap(arr[i], arr[largest]);

heapify(arr, N, largest);

}

}

void heapSort(int arr[], int N)

{

for (int i = N / 2 - 1; i >= 0; i--)

heapify(arr, N, i);

for (int i = N - 1; i > 0; i--) {

swap(arr[0], arr[i]);

heapify(arr, i, 0);

}

}

//Quick Sort

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

{

int pivot=arr[high];

int i=(low-1);

for(int j=low;j<=high;j++)

{

if(arr[j]<pivot)

{

i++;

swap(arr[i],arr[j]);

}

}

swap(arr[i+1],arr[high]);

return (i+1);

}

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

{

if(low<high)

{

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

quickSort(arr,low,pi-1);

quickSort(arr,pi+1,high);

}

}

// Merge sort

void merge(int array[], int const left, int const mid,

int const right)

{

int const subArrayOne = mid - left + 1;

int const subArrayTwo = right - mid;

// Create temp arrays

auto \*leftArray = new int[subArrayOne],

\*rightArray = new int[subArrayTwo];

// Copy data to temp arrays leftArray[] and rightArray[]

for (auto i = 0; i < subArrayOne; i++)

leftArray[i] = array[left + i];

for (auto j = 0; j < subArrayTwo; j++)

rightArray[j] = array[mid + 1 + j];

auto indexOfSubArrayOne = 0, indexOfSubArrayTwo = 0;

int indexOfMergedArray = left;

// Merge the temp arrays back into array[left..right]

while (indexOfSubArrayOne < subArrayOne

&& indexOfSubArrayTwo < subArrayTwo) {

if (leftArray[indexOfSubArrayOne]

<= rightArray[indexOfSubArrayTwo]) {

array[indexOfMergedArray]

= leftArray[indexOfSubArrayOne];

indexOfSubArrayOne++;

}

else {

array[indexOfMergedArray]

= rightArray[indexOfSubArrayTwo];

indexOfSubArrayTwo++;

}

indexOfMergedArray++;

}

// Copy the remaining elements of

// left[], if there are any

while (indexOfSubArrayOne < subArrayOne) {

array[indexOfMergedArray]

= leftArray[indexOfSubArrayOne];

indexOfSubArrayOne++;

indexOfMergedArray++;

}

// Copy the remaining elements of

// right[], if there are any

while (indexOfSubArrayTwo < subArrayTwo) {

array[indexOfMergedArray]

= rightArray[indexOfSubArrayTwo];

indexOfSubArrayTwo++;

indexOfMergedArray++;

}

delete[] leftArray;

delete[] rightArray;

}

// begin is for left index and end is right index

// of the sub-array of arr to be sorted

void mergeSort(int array[], int const begin, int const end)

{

if (begin >= end)

return;

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

mergeSort(array, begin, mid);

mergeSort(array, mid + 1, end);

merge(array, begin, mid, end);

}

int main()

{

// int arr[] = {9, 14, 10, 5};

int arr[] = {5, 1, 2, 3, 8, 6, 23, 10};

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

mergeSort(arr, 0, n-1);

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

{

cout << arr[i] << " ";

}

return 0;

}